УДК 519.254, 004.048 **DOI: 10.14529/cmse240404**

ОБ ОДНОЙ ФУНКЦИИ ПОТЕРЬ ДЛЯ ОБУЧЕНИЯ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ ВОССТАНОВЛЕНИЯ ВРЕМЕННЫХ РЯДОВ

© 2024 А.А. Юртин

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

Поступила в редакцию: 01.09.2024

В статье рассмотрена проблема выбора функции потерь для обучения нейросетевых моделей восстановления пропущенных значений многомерных временных рядов и предложена новая функция потерь, названная MPDE (Mean Profile Distance Error, средняя ошибка профиля расстояния). MPDE для истинной и восстановленной подпоследовательностей ряда, имеющих длину m, вычисляется как среднее значение расстояний между всеми парами окон (непрерывных промежутков) этих подпоследовательностей, имеющими длину ℓ, где $\ell \leqslant m$ и окна имеют одинаковые начальные индексы. Расстояние между двумя окнами представляет собой модификацию меры схожести MPdist (расстояние матричного профиля) и определяется как взвешенная сумма евклидова и z-нормированного евклидова расстояний между данными окнами. Веса слагаемых берутся из отрезка [0,1] и являются параметрами функции потерь. Функция MPDE позволяет при обучении нейросетевой модели учитывать поведенческое сходство сравниваемых подпоследовательностей, учитывая наличие в них сходных окон независимо от мест взаимного расположения этих окон. Функция потерь MPDE имеет высокую вычислительную сложность, поэтому для ее внедрения в фреймворки глубокого обучения разработан параллельный алгоритм, вычисляющий MPDE на графическом процессоре. Алгоритм реализован с помощью фреймворка PyTorch, который позволяет имплементировать MPDE как последовательность автоматически распараллеливаемых операций с многомерными тензорами. Эксперименты на многомерных временных рядах из различных предметных областей показали, что в 78% случаев передовые нейросетевые модели достигают наиболее высокой точности восстановления (по метрике RMSE) при использовании предложенной функции потерь; при этом модели демонстрируют точность восстановления на 40% выше среднего значения, достигнутого при использовании других функций потерь.

Ключевые слова: временной ряд, восстановление пропущенных значений, функция потерь, нейронные сети, PyTorch.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Юртин А.А. Об одной функции потерь для обучения нейросетевых моделей восстановления временных рядов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 4. С. 53–73. DOI: 10.14529/cmse240404.

Введение

В настоящее время интеллектуальная обработка временных рядов играет важную роль в различных предметных областях: экономика [1], энергетика [2], медицина [3], метеорология [4] и др. Одной из ключевых проблем интеллектуального анализа временных рядов является наличие в них пропущенных значений, которые могут возникать из-за неисправности сенсоров, потерь данных в процессе передачи или человеческого фактора. Пропуски в данных существенно затрудняют интеллектуальную обработку временных рядов, поскольку соответствующие методы и алгоритмы, как правило, требуют полной и непрерывной последовательности значений. За последние годы значительные успехи в области разработки эффективных методов восстановления пропущенных значений во временных рядах достигнуты благодаря применению нейросетевых моделей [5, 6]. Современные методы вос-

становления временных рядов способны учитывать временны́е зависимости и сложные поведенческие шаблоны в данных, повышая тем самым точность восстановления, на основе применения разнообразных нейросетевых архитектур: рекуррентные сети [7, 8], автоэнкодеры [9], трансформеры [10, 11] и др. При применении в восстановлении временных рядов нейросетевых моделей одним из ключевых факторов, влияющих на точность, является функция потерь (loss function) [12]. Функция потерь позволяет вычислить оценку несоответствия восстановленных и истинных значений. К функции потерь, как правило, предъявляются следующие требования [13, 14]: дифференцируемость и эффективность вычисления. *Дифференцируемость* необходима для применения метода градиентного спуска [15], который использует частные производные функции потерь для минимизации и обновления параметров нейросетевой модели. *Вычислительная эффективность* подразумевает быстрое вычисление функции при обучении нейросетевой модели на большом объеме данных. Кроме того, функция потерь выбирается, как правило, с учетом особенностей решаемой задачи, обучающей выборки и назначения нейросетевой модели. Например, в задачах обработки текста, как правило, используют функцию кросс-энтропии, которая позволяет нейросетевой модели оценить разницу между предсказанной и реальной вероятностями появления определенного слова на каждой позиции в предложении [16, 17]. В нейросетевой модели YOLO [18], используемой для обнаружения объектов на изображениях и видео, применяется составная функция потерь, компоненты которой учитывают расположение, класс и фон объекта.

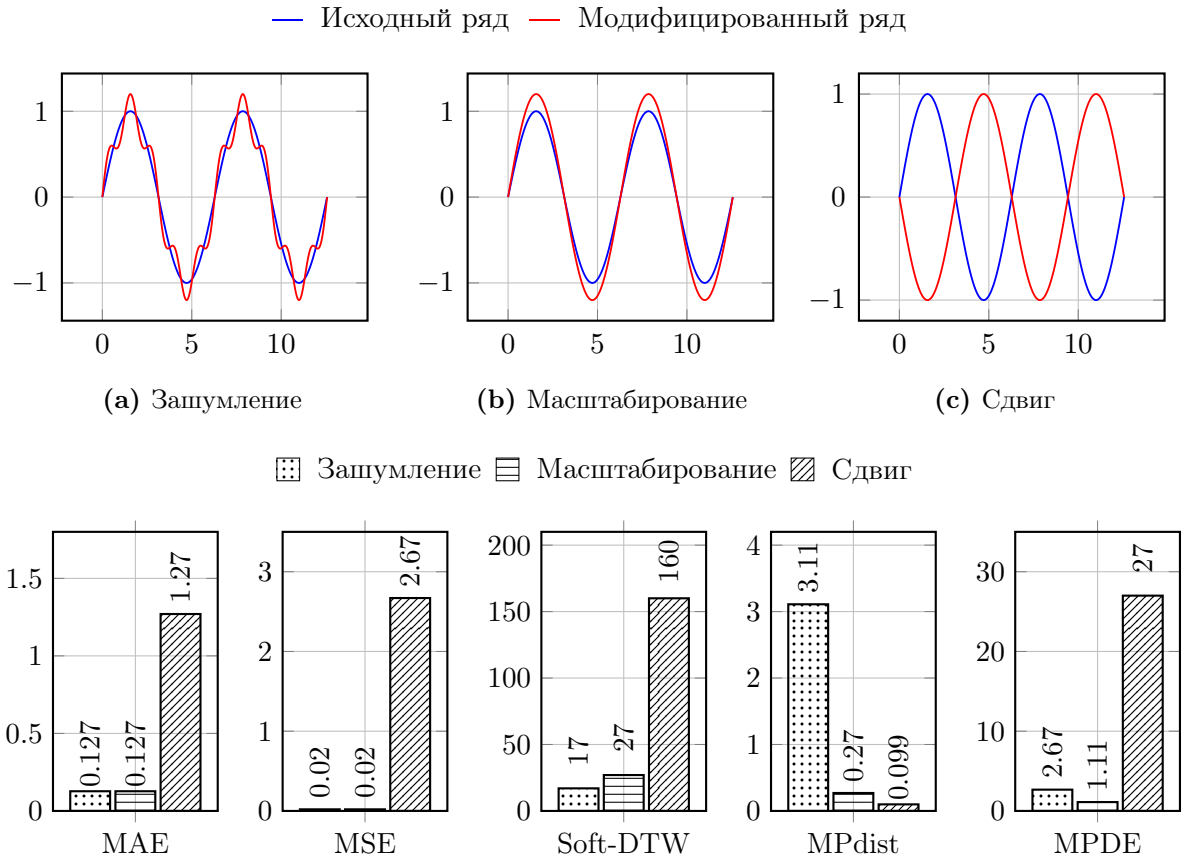
В контексте задачи восстановления пропущенных значений во временных рядах вышеуказанные требования к функции потерь могут быть дополнены, чтобы учитывать при сравнении истинной и восстановленной подпоследовательностей ряда следующие аспекты: поведенческая схожесть, схожесть по форме и схожесть абсолютных значений. *Поведенческое сходство* предполагает, что при сравнении подпоследовательностей учитывается наличие в них сходных непрерывных промежутков меньшей длины независимо от мест взаимного расположения этих промежутков. Другими словами, промежутки в сравниваемых подпоследовательностях могут отражать сходные активности субъектов, не совпадающие по времени. *Сходство по форме* предполагает, что сравниваемые подпоследовательности имеют близкие геометрическую структуру, амплитуду и др. Наконец, *сходство по абсолютным значениям* предполагает тождественность или минимальную разницу между подпоследовательностями при сравнении их соответствующих точек.

В данной работе предлагается новая функция потерь для обучения нейросетевых моделей, предназначенных для восстановления пропущенных значений во временных рядах, получившая название MPDE (Mean Profile Distance Error, средняя ошибка профиля расстояния). MPDE позволяет учитывать поведенческое сходство, сходство по форме и абсолютным значениям при сравнении истинной и восстановленной подпоследовательностей ряда, повышая тем самым точность восстановления. Для эффективного использования предложенной функции потерь, реализован параллельный алгоритм вычисления на фреймворке PyTorch [19].

Статья организована следующим образом. В разделе 2 приводятся формальные определения понятий и нотация, используемые в статье. Предложенная функция потерь и ее параллельная реализация описаны в разделе 3. Результаты вычислительных экспериментов, исследующих эффективность предложенного метода, приведены в разделе 4. Заключение содержит сводку полученных результатов и направления будущих исследований.

1. Обзор связанных работ

Для обучения нейросетевых моделей восстановления пропущенных значений во временных рядах используются, как правило, те же функции потерь, что и при решении задачи регрессии временных рядов (предсказание значения ряда на основе последовательности его предыдущих значений) [12, 13, 20]. Одними из наиболее часто используемых являются следующие функции потерь: средняя абсолютная ошибка (Mean Absolute Error, MAE) [21], функции потерь на базе среднеквадратичной ошибки (Mean Squared Error, MSE; Root Mean Squared Error, RMSE; Mean Squared Logarithmic Error, MSlE и др.), квантильная ошибка (Quantile Loss) [22], функция потерь на основе логарифма гиперболического косинуса (Log-cosh Loss) [23], функция потерь Хьюбера (Huber Loss) [24] и др. Указанные функции ориентированы на установление соответствия абсолютных значений [12, 13] и не учитывают сходство по форме и поведенческую схожесть. Однако в последнее время исследователями предлагаются функции потерь, позволяющие учитывать форму и поведенческую схожесть подпоследовательностей временных рядов. Например, в работе [25] предложена функция потерь Soft-DTW, основанная на динамической трансформации времени (DTW, Dynamic Time Warping) [26], которая позволяет учитывать временны́е искажения между подпоследовательностями. В работе [27] предложена функция MPdist для вычисления расстояния между подпоследовательностями временного ряда, учитывающая поведенческую схожесть.



(a) Зашумление **(b)** Масштабирование **(c)** Сдвиг

(d) Значения функций потерь

Рис. 1. Значения функций потерь, вычисленные после модификации временного ряда

Рассмотрим, как изменяется значение функций потерь (упомянутых выше и функции MPDE, предложенной в данной работе) в различных ситуациях изменения данных временного ряда (см. рис. 1). Пусть имеется временной ряд, представляющий собой значения синусоидына отрезке $[0, 15]$ взятые с шагом 0.025, к точкам которого применяются три оператора, имитирующие различные ситуации, могущие возникать в процессе обучения нейросетевых моделей восстановления временных рядов: зашумление, масштабирование и сдвиг. Оператор зашумления предполагает добавление к точкам ряда значений другой синусоиды из того же отрезка, имеющей амплитуду 0.02. Оператор масштабирования умножает значение каждой точки ряда на 1.2. Оператор сдвига изменяет фазу на π. Результат действия указанных операторов над имеющимся рядом представлен на рис. 1a, 1b и 1c соответственно. Применение оператора зашумления приводит к отсутствию всех рассмотренных выше видов схожести между исходным и результирующим рядами: поведенческое, по форме и по абсолютным значениям. Оператор масштабирования приводит к потере схожести по абсолютным значениям и форме (последняя — частично), поведенческое сходство сохраняется. Наконец, оператор сдвига сохраняет поведенческую схожесть, но нарушает сходство по абсолютным значениям и форме.

Однако, можно видеть (см. рис. 1d), что результаты вычисления функций потерь для исходного и трансформированного рядов не вполне совпадают с отмеченными выше наблюдениями. Функции потерь MSE и MAE для случаев применения операторов зашумления и масштабирования дают одинаковую ошибку. Функция Soft-DTW оценивает случай внедрения шума в данные как более близкий к истине, чем случай применения масштабирования. Функция MPdist верно оценивает масштабирование как более близкий к истине случай, чем зашумление. Однако в силу своей природы (оценка поведенческого сходства между рядами) MPdist определяет наиболее близким к истине случаем применение сдвига, что некорректно в случае задачи восстановления временного ряда. Можно заключить, что рассмотренные выше функции потерь не учитывают одновременно все рассмотренные выше аспекты схожести подпоследовательностей временного ряда, важные в задаче восстановления пропущенных значений: поведенческое сходство, сходство по форме и сходство по абсолютным значениям. Далее в данной работе предлагается функция потерь, которая нацелена на преодоление указанных ограничений.

2. Теоретический базис

Одномерный временной ряд представляет собой хронологически упорядоченную последовательность вещественных значений:

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

Длина временного ряда, n, обозначается как $|T|$.

Подпоследовательность $T_{i,\,m}$ одномерного временного ряда T — это непрерывный промежуток из m элементов ряда, начиная с i-го элемента:

$$T_{i,\,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 1 \leqslant i \leqslant n - m + 1, \quad 3 \leqslant m \ll n. \tag{2}$$

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначим как S_T^m. Рассмотрим подпоследовательность $T_{i,\,m}$ как самостоятельный временной ряд. Тогда его подпоследовательность, имеющую заданную длину ℓ (где $\ell < m$), будем называть *ок-*

ном. Мощность множества окон заданной подпоследовательности обозначим соответственно как c, т.е. $c = |S^{\ell}_{T_{i,\,m}}| = m - \ell + 1$.

Многомерный временной ряд — это набор семантически связанных одномерных временных рядов одинаковой длины, которые синхронизированы во времени. Пусть d обозначает *размерность* многомерного ряда ($d > 1$), количество *измерений* — одномерных рядов в нем. Подобно одномерному случаю, многомерный временной ряд, его подпоследовательность и отдельные точки обозначим как $\boldsymbol{T}$, $\boldsymbol{T}_{i,\,m}$ и $\boldsymbol{t}_i$ соответственно, и определим их следующим образом:

$$\boldsymbol{T} = [\{T^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{3}$$

$$\boldsymbol{T}_{i,\,m} = [\{T^{(k)}_{i,\,m}\}_{k=1}^{d}]^{\intercal}, \tag{4}$$

$$\boldsymbol{t}_i = [\{t^{(k)}_{i}\}_{k=1}^{d}]^{\intercal}. \tag{5}$$

Для вычисления расстояний между окнами подпоследовательности в данной работе используются евклидово расстояние и его z-нормализованная версия, обозначаемые как $\mathrm{ED}(\cdot,\cdot)$ и $\widehat{\mathrm{ED}}(\cdot,\cdot)$ соответственно, и вычисляемые следующим образом:

$$\mathrm{ED}(X,\,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{6}$$

$$\widehat{\mathrm{ED}}(X,\,Y) = \mathrm{ED}(\widehat{X},\,\widehat{Y}), \tag{7}$$

$$\widehat{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \mu_x = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i, \quad \sigma_x = \sqrt{\frac{1}{\ell - 1}\sum_{i=1}^{\ell}(x_i - \mu_x)^2 + \varepsilon}, \tag{8}$$

где ε представляет собой машинный эпсилон, предотвращающий деление на ноль в вычислениях градиентов [28].

3. Функция потерь на основе среднего расстояния профилей

3.1. Формальное определение функции потерь

Предлагаемая функция потерь основана на средней ошибке профиля расстояния (MPDE, Mean Profile Distance Error) и вычисляется между двумя подпоследовательностями многомерного ряда $\boldsymbol{T}$, имеющими длину m: $\boldsymbol{X}$, $\boldsymbol{Y} \in S^{m}_{\boldsymbol{T}}$, где $\boldsymbol{X}$ играет роль восстановленных, а $\boldsymbol{Y}$ — реальных данных. Далее рассмотрим формальное определение функции MPDE для одномерного ряда (т.е. для случая X, $Y \in S^{m}_{T}$), а затем обобщим это определение на случай многомерного ряда.

MPDE вычисляется как среднее значение расстояний между всеми парами окон этих подпоследовательностей, имеющими длину ℓ, где $\ell \leqslant m$ и окна имеют одинаковые индексы. Расстояние между окнами вычисляется как взвешенная сумма двух слагаемых: евклидова расстояния между z-нормализованными версиями окон и евклидова расстояния между исходными данными. Веса указанных слагаемых, α и β соответственно, представляют собой вещественные числа из отрезка [0,1], и являются параметрами функции потерь. Формально MPDE может быть представлено следующим образом:

$$\begin{gathered}\mathrm{MPDE}(X,\,Y) = \frac{1}{c}\sum_{i=1}^{c}\alpha\mathrm{ED}(X_{i,\,i+\ell}, Y_{i,\,i+\ell}) + \beta\widehat{\mathrm{ED}}(X_{i,\,i+\ell}, Y_{i,\,i+\ell}),\\ 0 \leqslant \alpha,\ \beta \leqslant 1.\end{gathered} \tag{9}$$

Рассмотрим дифференцируемость функции MPDE. Частная производная MPDE по переменной x_i вычисляется как среднее значение производных расстояний между окнами, содержащими точку x_i:

$$\frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_i} = \frac{1}{c}\sum_{j=i-\frac{\ell}{2}}^{i+\frac{\ell}{2}} \frac{\partial \mathrm{ED}(X_{j,j+\ell}, Y_{j,j+\ell})}{\partial x_i} + \frac{\partial \widehat{\mathrm{ED}}(X_{j,j+\ell}, Y_{j,j+\ell})}{\partial x_i}, \tag{10}$$

Учитывая формул 6, 7 и 8, частная производная функции потерь по переменной x_i может быть записана в следующем виде:

$$\begin{gathered}\frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_i} = A + B + C,\\ A = \frac{\alpha}{c}\cdot\sum_{j=i-\frac{\ell}{2}}^{i+\frac{\ell}{2}} \frac{x_i - y_i}{\mathrm{ED}(X_{j,j+\ell}, Y_{j,j+\ell})},\ B = \frac{\beta}{c}\cdot\sum_{j=i-\frac{\ell}{2}}^{i+\frac{\ell}{2}} \frac{\widehat{x}_i - \widehat{y}_i}{\mathrm{ED}(X_{j,j+\ell}, Y_{j,j+\ell})},\\ C = \frac{\beta}{\ell\cdot c}\cdot\sum_{j=i-\frac{\ell}{2}}^{i+\frac{\ell}{2}} \frac{\partial \mathrm{ED}(X_{j,j+\ell}, Y_{j,j+\ell})}{\partial \mu}\cdot\frac{\partial \mu}{\partial x_i} + \frac{\beta}{\sigma\cdot c}\cdot\sum_{j=i-\frac{\ell}{2}}^{i+\frac{\ell}{2}} \frac{\partial \mathrm{ED}(X_{j,j+\ell}, Y_{j,j+\ell})}{\partial \sigma}\cdot\frac{\partial \sigma}{\partial x_i}.\end{gathered} \tag{11}$$

Рассмотрим слагаемые A, B и C в данной формуле более подробно. Слагаемое A включает в себя сумму расстояний между точками окон, что позволяет во время обучения нейросетевой модели восстановления ряда компенсировать различия в масштабах восстановленной и истинной подпоследовательностей. Слагаемое B задействует разность между нормализованными точками окон. Это дает возможность при обучении модели учесть сходство форм восстановленной и истинной подпоследовательностей. Наконец, слагаемое C включает в себя производные параметров нормализации μ и σ по переменной x_i и позволяет учитывать, как изменение одной точки влияет на параметры нормализации для всего окна. Изменения весов α и β позволяет регулировать вклады слагаемых A и B, C соответственно в итоговое значение функции потерь.

В случае многомерного ряда значение функции потерь от аргументов $\boldsymbol{X},\ \boldsymbol{Y} \in S^m_{\boldsymbol{T}}$ вычисляется как среднее арифметическое значений функции MPDE по всем измерениям между соответствующими одномерными подпоследовательностями $X^{(i)},\ Y^{(i)} \in S^m_{T^{(i)}}$:

$$\mathrm{MPDE}(\boldsymbol{X},\ \boldsymbol{Y}) = \frac{1}{d}\sum_{i=1}^{d} \mathrm{MPDE}(X^{(i)}, Y^{(i)}). \tag{12}$$

Очевидно, однако, что функция MPDE имеет существенно более высокую вычислительную сложность, чем известные аналоги, поскольку в ней необходимо вычисление евклидовых расстояний между всеми исходными и z-нормализованными окнами входных подпоследовательностей. Для решения данной проблемы и применения функции MPDE на практике для обучения нейросетевых моделей восстановления временных рядов в данной работе предлагается параллельный алгоритм вычисления данной функции потерь, который рассмотрен в следующем разделе.

3.2. Алгоритм вычисления функции потерь

Для параллельной реализации вычисления функции потерь, описанной в разделе 3.1, используется фреймворк PyTorch [19], который в настоящее время является де-факто стандартом реализации нейросетевых моделей. Основной структурой для хранения данных в

PyTorch является тензор (Tensor), представляющий собой многомерную матрицу вещественных значений. Фреймворк предоставляет разработчику набор базовых методов для работы с тензорами: вычисление среднего значения, квадратного корня, стандартного отклонения и др. Фреймворк также поддерживает автоматическое дифференцирование базовых методов и их параллельное исполнение на графических процессорах [29]. Для реализации используются базовые методы, выполняющие два вида операций: агрегация и извлечение данных. Методы агрегации выполняют над входным тензором одну из следующих операций: вычисление среднего `mean`, суммы `sum`, квадратного корня `sqrt` и дисперсии `var`. Агрегация может выполняться по всем измерениям либо вдоль заданного измерения, превращая размерность этого измерения в единицу. Извлечение данных реализуется с помощью метода `unfold`, который для заданного тензора возвращает все его подмассивы заданной длины, взятые вдоль заданного измерения.

Алг. 1 TorchMPDE(IN: $\boldsymbol{X}$, $\boldsymbol{Y} \in \mathbb{R}^{b\times m\times d}$, α, β; OUT: MPDE)

1: $\boldsymbol{X}_{\text{slice}} \leftarrow \texttt{unfold}(\boldsymbol{X}, \ell, 1)$; $\boldsymbol{Y}_{\text{slice}} \leftarrow \texttt{unfold}(\boldsymbol{Y}, \ell, 1)$
2: $\mu_x \leftarrow \texttt{mean}(\boldsymbol{X}_{\text{slice}}, 2)$; $\mu_y \leftarrow \texttt{mean}(\boldsymbol{Y}_{\text{slice}}, 2)$
3: $\sigma_x \leftarrow \texttt{sqrt}(\texttt{var}(\boldsymbol{X}_{\text{slice}}, 2) + \varepsilon)$; $\sigma_y \leftarrow \texttt{sqrt}(\texttt{mean}(\boldsymbol{Y}_{\text{slice}}, 2) + \varepsilon)$
4: $\boldsymbol{X}_{\text{norm}} \leftarrow (\boldsymbol{X}_{\text{slice}} - \mu_x)/\sigma_x$; $\boldsymbol{Y}_{\text{norm}} \leftarrow (Y_{\text{slice}} - \mu_y)/\sigma_y$
5: $\text{ED2}_{\text{norm}} \leftarrow \texttt{sum}((X_{\text{norm}} - \boldsymbol{Y}_{norm})^2, 3)$; $\text{ED2} \leftarrow \texttt{sum}((\mathbf{X}_{\text{slice}} - \boldsymbol{Y}_{\text{slice}})^2, 3)$
6: $\text{ED}_{\text{norm}} \leftarrow \texttt{sqrt}(\text{ED2}_{\text{norm}} + \varepsilon)$; $\text{ED} \leftarrow \texttt{sqrt}(\text{ED2} + \varepsilon)$
7: $\text{MPDE} \leftarrow \texttt{mean}(\alpha \cdot \text{ED} + \beta \cdot \text{ED}_{\text{norm}})/2$
8: **return** MPDE

Реализация вычисления функции потерь MPDE представлена в алг. 1. Для встраивания реализации во фреймворк глубокого обучения функция MPDE должна оперировать пакетами (batch), которые состоят из пар с восстановленным и истинным значением. Поэтому, помимо параметров α и β, входными данными алгоритма являются два массива длины b, в которых хранятся восстановленные и соответствующие им истинные многомерные под последовательности длины m, обозначенные как трехмерные тензоры $\boldsymbol{X}$, $\boldsymbol{Y} \in \mathbb{R}^{b\times m\times d}$ соответственно. На первом шаге алгоритма над пакетом подпоследовательностей выполняется операция `unfold` вдоль измерения времени. В результате каждая подпоследовательность разбивается на окна длины ℓ. Множества всех окон каждой подпоследовательности входных пакетов сохраняются в 4-мерные тензоры X_{slice}, $Y_{\text{slice}} \in \mathbb{R}^{b\times m-\ell\times\ell\times d}$. Далее для каждого окна с помощью операций `mean`, `var` и `sqrt` вычисляются промежуточные данные z-нормализации: средние значения μ_x, μ_y и среднеквадратичные отклонения σ_x, σ_y. Затем каждый тензор, хранящий окна подпоследовательностей, нормализуется с учетом данных, вычисленных на предыдущем шаге. После этого происходит вычисление евклидовых расстояний между двумя типами окон: нормализованными и исходными (см. строки 5 и 6 соответственно). На последнем шаге полученные расстояния умножаются на весовые коэффициенты α и β, а результаты усредняются по пакету. Полученный результат является итоговым значением функции потерь.

4. Вычислительные эксперименты

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты на оборудовании Лаборатории суперкомпьютерного моделирования

Южно-Уральского государственного университета (Челябинск) [30]. В экспериментах сравнивалась точность восстановления пропусков, которую показывают передовые нейросетевые модели, обученные с помощью предложенной функции MPDE и другими общепринятыми функциями потерь. Сравнение осуществлялось с использованием наборов данных из различных предметных областей. Кроме того, в экспериментах исследовалось быстродействие вычисления указанных функций потерь.

4.1. Описание экспериментов

В экспериментах выполнялось сравнение предложенной функции потерь со следующими функциями потерь: средняя абсолютная ошибка (Mean Absolute Error, MAE), среднеквадратичная ошибка (Mean Squared Error, MSE) и квантильная функция потерь (Quantile Loss) [22].

Таблица 1. Наборы данных, используемые в экспериментах

№	Набор	Длина, $n \times 10^3$	Количество, измерений, d	Предметная область
Группа А: Активность субъекта				
1.	Electricity [31]	5	9	Потребление электроэнергии в нескольких домашних хозяйствах
2.	Madrid [32]	25	10	Трафик автомобильных дорог в Мадриде
3.	NREL [33]	8.7	9	Потребление электроэнергии лаборатории в США
4.	PAMAP [34]	50	10	Показания носимого датчика во время активности человека
5.	WalkRun [35]	37	11	
Группа Б: Сезонность и циклы				
6.	BAFU [36]	50	10	Сброс воды в реках Швейцарии
7.	Climate [37]	5	10	Погода в различных локациях Северной Америки
8.	MeteoSwiss [38]	10	10	Погода в городах Швейцарии
9.	Saaleaue [39]	23	14	Погода в городах Германии

Оценка предложенного метода и его сравнения с аналогами проводилась с использованием временных рядов, резюмированных в табл. 1. Временные ряды могут быть разделены на две группы в соответствии с особенностями данных: в группу А входят ряды 1–5, для которых характерно отсутствие сезонности и цикличности компонентов ввиду возможности случайного изменения активности субъекта; группа Б включает в себя ряды 6–9, которые демонстрируют сезонность и цикл (циклические изменения уровня ряда с постоянным и переменным периодом соответственно). Пропуски в многомерном временном ряде формировались в соответствии с наиболее тяжелым для восстановления сценарием Blackout [40], который предполагает пустые значения в каждой из 100 последних точек в каждом измерении многомерного временного ряда.

Для оценки точности восстановления в данной работе используется мера корня из среднеквадратичной ошибки RMSE (Root Mean Square Error) как одна из наиболее часто применяемых для этих целей метрик [41], определяемая следующим образом:

$$\mathrm{RMSE} = \sqrt{\frac{1}{h}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}, \tag{13}$$

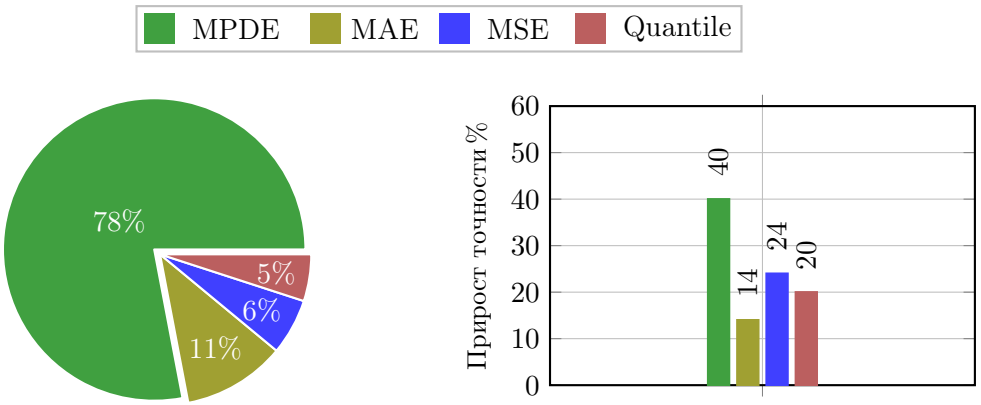
где y_i — фактическое значение, $\hat{y}_i$ — восстановленное значение, h — количество восстановленных точек.

Для оценки предложенной в работе функции потерь использовались методы, основанные на различных архитектурах: рекуррентные (BRITS [7] и SANNI [42]), автоэнкодеры (SAETI [35]) и трансформеры (SAITS [10]) В табл. 2 резюмированы параметры обучения нейросетевых моделей, используемых для оценки предложенного метода. Оптимальные параметры функции потерь MPDE для каждой комбинации $\langle$*Модель*, *Набор данных*$\rangle$ были подобраны с помощью решетчатого поиска [43] в диапазоне $[0.1, 0.9]$ с шагом 0.1.

Таблица 2. Параметры обучения, используемые в экспериментах

№	Название	Значение
1.	Количество эпох (Epochs)	100
2.	Скорость обучения (Learning rate)	5×10^{-3}
3.	Оптимизатор (Optimizer)	Adam
4.	Размер пакета (Batch size)	256
5.	Длина подпоследовательности (m)	100
6.	Длина окна (ℓ)	50

4.2. Анализ результатов



(a) Доля экспериментов с лучшей точностью

(b) Средний прирост точности при использовании функций потерь

Рис. 2. Точность моделей при использовании различных функций потерь

На рис. 2 представлена доля пар $\langle$*Модель*, *Набор данных*$\rangle$ в которых каждая функция потерь продемонстрировала лучший результат. На рис. 2b показан средний выигрыш (среднее повышение точности в процентах) лучшей функции потерь у прочих функций. Подробные результаты точности восстановления, полученной каждой функцией потерь на каждом наборе данных, разбитые для удобства просмотра по группам данных (А — активность субъекта и Б — сезонность и циклы), представлены на рис. 4 и 5 соответственно.

Можно видеть, что предложенная функция потерь MPDE позволяет минимум в 78% случаев повысить точность восстановления минимум на 40% по сравнению с рассмотренными передовыми функциями потерь. В случае, когда применение MPDE не дает наивысшую точность восстановления, ее отставание от лучшей функции потерь составляет не более 6%.

Отдельно по группам наборов данных MPDE в среднем опережает аналоги по точности на 34% и 46% и является лучшей в 80% и 75% случаев в группах А и Б соответственно.

Таблица 3. Оптимальные параметры MPDE

№	Название	Модель							
		BRITS		**SANNI**		**SAETI**		**SAITS**	
		α	β	α	β	α	β	α	β
Группа А: Активность субъекта									
1.	Electricity	0.7	0.2	0.6	0.1	0.1	0.8	0.5	0.5
2.	Madrid	0.2	0.7	0.5	0.3	0.2	0.5	0.8	0.5
3.	NREL	0.6	0.1	0.5	0.3	0.8	0.2	0.2	0.5
4.	PAMAP	0.4	0.2	0.4	0.1	0.5	0.6	0.4	0.1
5.	WalkRun	0.8	0.4	0.1	0.2	0.3	0.3	0.8	0.2
Группа Б: Сезонность и циклы									
6.	BAFU	0.9	0.1	0.1	0.1	0.2	0.7	0.7	0.2
7.	Climate	0.4	0.9	0.4	0.3	0.5	0.3	0.3	0.8
8.	MeteoSwiss	0.3	0.1	0.2	0.6	0.7	0.8	0.0	0.8
9.	Saaleaue	0.1	0.1	0.6	0.3	0.9	0.4	0.1	0.7

В табл. 3 приведены значения параметров α и β функции MPDE, при которых нейросетевая модель достигает наиболее высокой точности восстановления. Средние значения весовых коэффициентов α и β — по всем наборам данных (0.47 и 0.34) или отдельно для групп данных А (0.4 и 0.45) и Б (0.44 и 0.39) — могут быть использованы как начальные значения для настройки этих гиперпараметров — в случае, когда характеристики восстанавливаемого ряда неизвестны или ряд можно причислить к одной из указанных выше групп соответственно. Автоматизация подбора гиперпараметров α и β может рассматриваться как одно из направлений будущих исследований.

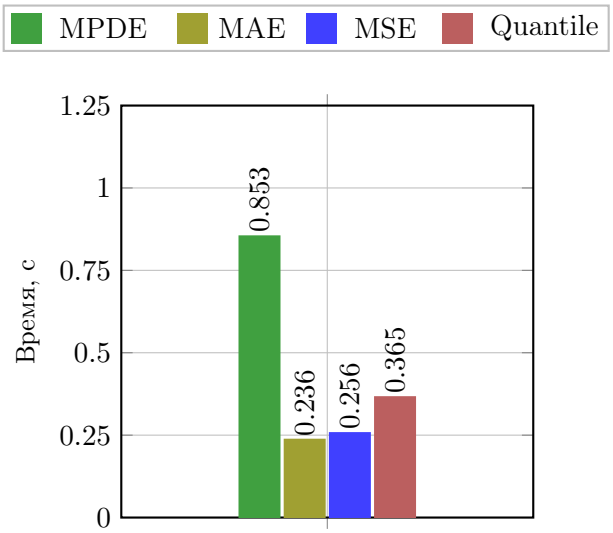


Рис. 3. Быстродействие вычисления функции потерь

На рис. 3 представлено сравнение быстродействия вычисления рассмотренных в описанных выше экспериментах функций потерь. В данном эксперименте сравнивалось время, за которое вычислялись функции потерь для пакета из 128 многомерных подпоследовательностей длиной $m = 100$ (для функции MPDE длина окна $\ell = 50$) и количеством измерений $d = 10$, усредненное по 10 тыс. запусков. Можно видеть, что вычисление MPDE происходит в среднем в 4 раза медленнее, чем у известных функций потерь. Однако это необходимая плата за более высокую точность восстановления, которую обеспечивает функция MPDE, по сравнению с известными функциями потерь.

Заключение

В данной статье затронута проблема повышения точности нейросетевых моделей восстановления пропущенных значений в многомерных временных рядах, которая является актуальной в широком спектре предметных областей. Предложена новая функция потерь для обучения нейросетевых моделей восстановления временных рядов, которая получила название MPDE (Mean Profile Distance Error) и основана на средней ошибке профиля расстояния. MPDE для истинной и восстановленной подпоследовательностей ряда вычисляется как среднее значение расстояний между всеми парами окон (непрерывных промежутков) этих подпоследовательностей, имеющими меньшую длину, где окна имеют одинаковые начальные индексы. Расстояние между двумя окнами определяется как взвешенная сумма евклидова и z-нормированного евклидова расстояний между данными окнами. Веса слагаемых берутся из отрезка [0,1] и являются гиперпараметрами функции потерь, равно как и длины подпоследовательности и окна. При обучении нейросетевой модели и многократном сравнении истинных и восстановленных подпоследовательностей ряда MPDE учитывает поведенческое сходство, сходство по форме и абсолютным значениям, повышая тем самым точность восстановления. Поведенческое сходство предполагает, что при сравнении подпоследовательностей учитывается наличие в них сходных окон независимо от мест взаимного расположения этих окон. Другими словами, MPDE учитывает, что в сравниваемых подпоследовательностях окна могут отражать сходные активности субъекта, не совпадающие по времени.

Функция MPDE имеет существенно более высокую вычислительную сложность, чем известные аналоги, поскольку в ней необходимо вычисление евклидовых расстояний между всеми исходными и z-нормализованными окнами входных подпоследовательностей. Для эффективного использования предложенной функции потерь в обучении нейросетевых моделей реализован параллельный алгоритм вычисления MPDE, работа которого кратко может быть описана следующим образом. Сначала каждый входной пакет подпоследовательностей разбивается на окна. Далее для каждого окна вычисляются параметры z-нормализации (среднее и стандартное отклонение), после чего выполняется нормализация всех окон. Затем алгоритм вычисляет евклидовы расстояния между нормализованными и исходными окнами. Наконец, полученные расстояния с учетом весов усредняются по пакету. Алгоритм реализован на базе фреймворка PyTorch [19], который инкапсулирует параллелизм вычислений на графическом процессоре. В вычислительных экспериментах сравнивалась точность восстановления, которую демонстрируют передовые нейросетевые модели на многомерных временных рядах из различных предметных областей, при применении MPDE и других известных функций потерь. Результаты экспериментов показывают, что в 78% случаев нейросетевые модели достигают наиболее высокой точности восстановления (по метрике

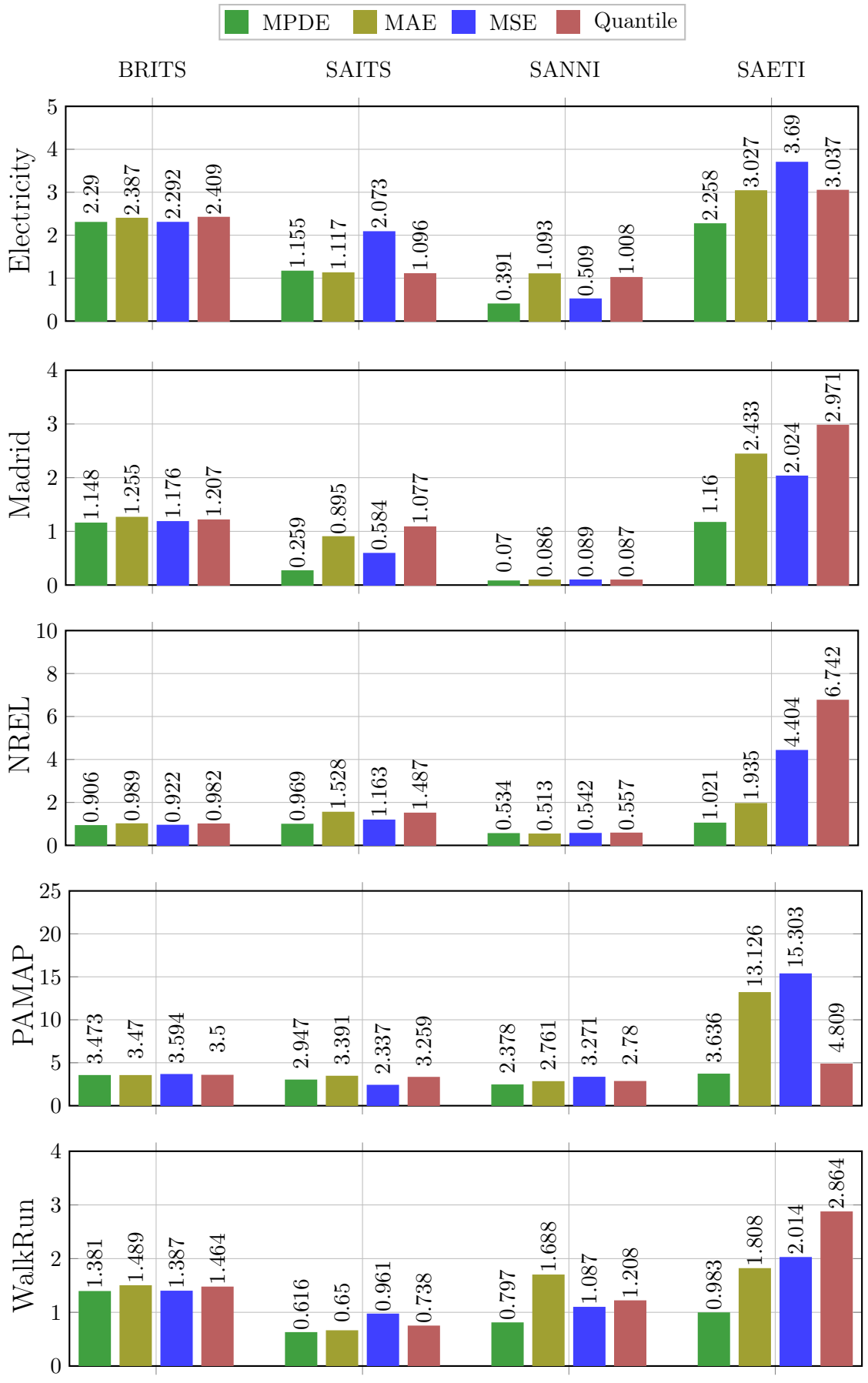


Рис. 4. Результаты экспериментов для данных группы A (RMSE)

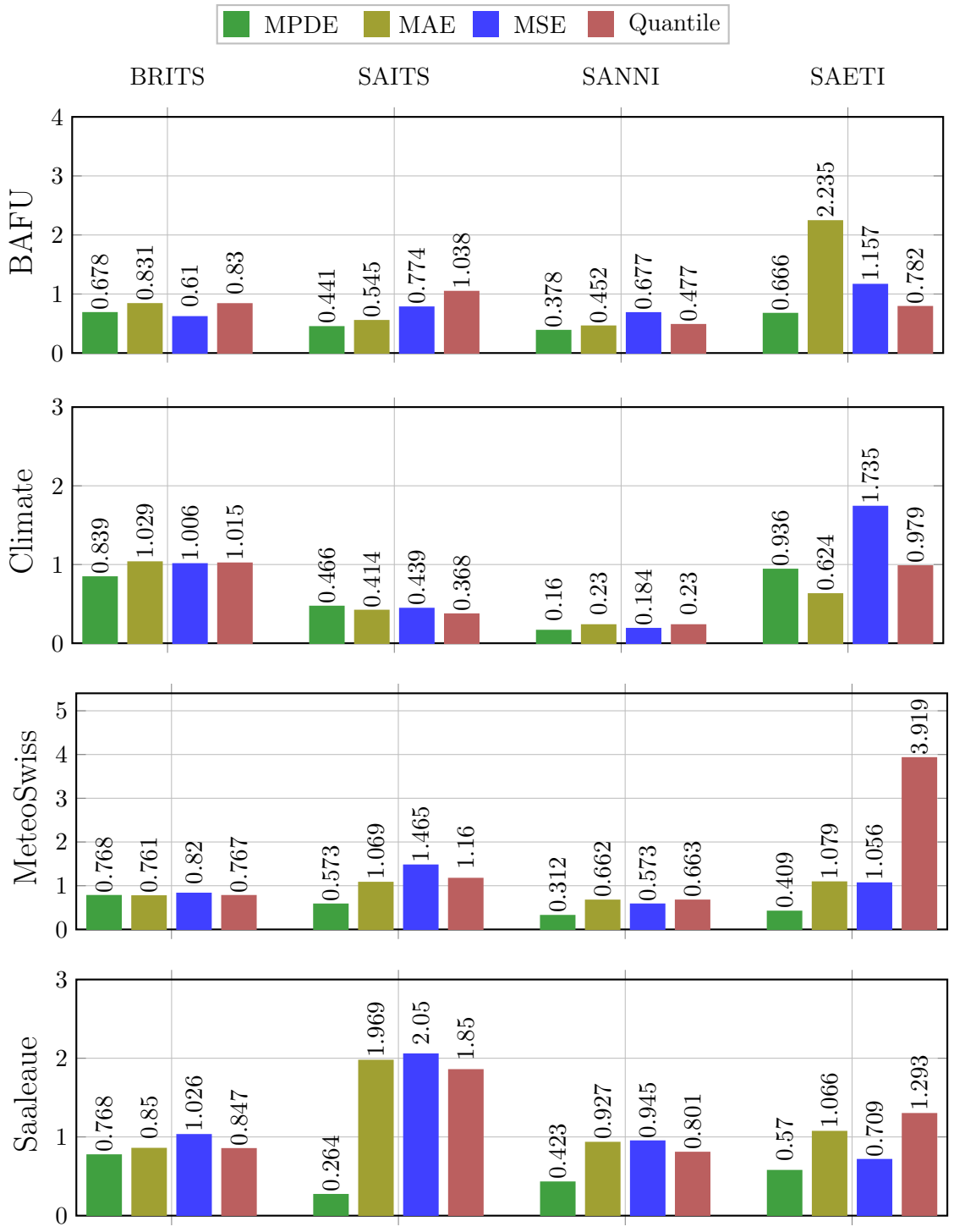


Рис. 5. Результаты экспериментов для данных группы Б (RMSE)

RMSE) при использовании предложенной функции потерь; при этом точность восстановления на 40% выше среднего значения, достигнутого при использовании других функций потерь. Однако, необходимой платой за более высокую точность восстановления является более низкое быстродействие вычисления MPDE по сравнению с известными функциями потерь (в среднем в 4 раза медленнее).

В будущих исследованиях планируется рассмотреть возможность автоматизации процесса подбора гиперпараметров функции потерь.

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

Литература

1. Majumdar S., Laha A.K. Clustering and classification of time series using topological data analysis with applications to finance // Expert Syst. Appl. 2020. Vol. 162. P. 113868. DOI: 10.1016/j.eswa.2020.113868.
2. Lara-Benıtez P., Carranza-Garcıa M., Luna-Romera J.M., Riquelme J.C. Temporal convolutional networks applied to energy-related time series forecasting // applied sciences. 2020. Vol. 10, no. 7. P. 2322. DOI: 10.3390/app10072322.
3. Gratius N., Wang Z., Hwang M.Y., *et al.* Digital Twin Technologies for Autonomous Environmental Control and Life Support Systems // J. Aerosp. Inf. Syst. 2024. Vol. 21, no. 4. P. 332–347. DOI: 10.2514/1.I011320.
4. Zhou Z., Tang W., Li M., *et al.* A Novel Hybrid Intelligent SOPDEL Model with Comprehensive Data Preprocessing for Long-Time-Series Climate Prediction // Remote. Sens. 2023. Vol. 15, no. 7. P. 1951. DOI: 10.3390/RS15071951.
5. Kazijevs M., Samad M.D. Deep imputation of missing values in time series health data: A review with benchmarking // J. Biomed. Informatics. 2023. Vol. 144. P. 104440. DOI: 10.1016/J.JBI.2023.104440.
6. Fang C., Wang C. Time Series Data Imputation: A Survey on Deep Learning Approaches // CoRR. 2020. Vol. abs/2011.11347. arXiv: 2011.11347. URL: https://arxiv.org/abs/2011.11347.
7. Cao W., Wang D., Li J., *et al.* BRITS: Bidirectional Recurrent Imputation for Time Series // Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada / ed. by S. Bengio, H.M. Wallach, H. Larochelle, *et al.* 2018. P. 6776–6786. URL: https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html.
8. Yoon J., Zame W.R., Schaar M. van der Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks // IEEE Trans. Biomed. Eng. 2019. Vol. 66, no. 5. P. 1477–1490. DOI: 10.1109/TBME.2018.2874712.
9. Fortuin V., Baranchuk D., Rätsch G., Mandt S. GP-VAE: Deep Probabilistic Time Series Imputation // The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Vol. 108 / ed. by S. Chiappa, R. Calandra. PMLR, 2020. P. 1651–1661. Proceedings of Machine Learning Research. URL: http://proceedings.mlr.press/v108/fortuin20a.html.
10. Du W., Côté D., Liu Y. SAITS: Self-attention-based imputation for time series // Expert Syst. Appl. 2023. Vol. 219. P. 119619. DOI: 10.1016/J.ESWA.2023.119619.
11. Oh E., Kim T., Ji Y., Khyalia S. STING: Self-attention based Time-series Imputation Networks using GAN. 2021. DOI: 10.1109/ICDM51629.2021.00155.
12. Kaya M., Bilge H.S. Deep Metric Learning: A Survey // Symmetry. 2019. Vol. 11, no. 9. P. 1066. DOI: 10.3390/SYM11091066.

13. Wang Q., Ma Y., Zhao K., Tian Y. A comprehensive survey of loss functions in machine learning // Annals of Data Science. 2022. Vol. 9. P. 187–212. DOI: 10.1007/s40745-020-00253-5.

14. Ciampiconi L., Elwood A., Leonardi M., *et al.* A survey and taxonomy of loss functions in machine learning // CoRR. 2023. Vol. abs/2301.05579. DOI: 10.48550/ARXIV.2301.05579. arXiv: 2301.05579.

15. Netrapalli P. Stochastic gradient descent and its variants in machine learning // Journal of the Indian Institute of Science. 2019. Vol. 99, no. 2. P. 201–213. DOI: 10.1007/s41745-019-0098-4.

16. Xu J., Ren X., Lin J., Sun X. Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018 / ed. by E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii. Association for Computational Linguistics, 2018. P. 3940–3949. DOI: 10.18653/V1/D18-1428.

17. Chang Y., Wang X., Wang J., *et al.* A Survey on Evaluation of Large Language Models // ACM Trans. Intell. Syst. Technol. 2024. Vol. 15, no. 39. P. 1–45. DOI: 10.1145/3641289.

18. Chen W., Huang H., Peng S., *et al.* YOLO-face: a real-time face detector // Vis. Comput. 2021. Vol. 37, no. 4. P. 805–813. DOI: 10.1007/S00371-020-01831-7.

19. Paszke A., Gross S., Massa F., *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library // Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada / ed. by H.M. Wallach, H. Larochelle, A. Beygelzimer, *et al.* 2019. P. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

20. Jadon A., Patil A., Jadon S. A Comprehensive Survey of Regression-Based Loss Functions for Time Series Forecasting // International Conference on Data Management, Analytics & Innovation. Springer. 2024. P. 117–147. DOI: 10.1007/978-981-97-3245-6_9.

21. Qi J., Du J., Siniscalchi S.M., *et al.* On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression // IEEE Signal Process. Lett. 2020. Vol. 27. P. 1485–1489. DOI: 10.1109/LSP.2020.3016837.

22. Chen X., Liu W., Mao X., Yang Z. Distributed High-dimensional Regression Under a Quantile Loss Function // J. Mach. Learn. Res. 2020. Vol. 21, no. 182. P. 1–43. URL: http://jmlr.org/papers/v21/20-297.html.

23. Saleh R.A., Saleh A.K.M.E. Statistical Properties of the log-cosh Loss Function Used in Machine Learning // CoRR. 2022. Vol. abs/2208.04564. DOI: 10.48550/ARXIV.2208.04564. arXiv: 2208.04564.

24. Sun Q., Zhou W.-X., Fan J. Adaptive Huber regression // Journal of the American Statistical Association. 2020. Vol. 115, no. 529. P. 254–265. DOI: 10.1080/01621459.2018.1543124.

25. Cuturi M., Blondel M. Soft-DTW: a Differentiable Loss Function for Time-Series // Proceedings of the 34th International Conference on Machine Learning. Vol. 70 / ed. by D. Precup, Y.W. Teh. PMLR, June 2017. P. 894–903. Proceedings of Machine Learning Research. URL: https://proceedings.mlr.press/v70/cuturi17a.html.

26. Berndt D.J., Clifford J. Using Dynamic Time Warping to find patterns in time series // KDD Workshop. 1994. P. 359–370. URL: https://cdn.aaai.org/Workshops/1994/WS-94-03/WS94-03-031.pdf.

27. Gharghabi S., Imani S., Bagnall A.J., *et al.* Matrix Profile XII: MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios // IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018. IEEE Computer Society, 2018. P. 965–970. DOI: 10.1109/ICDM.2018.00119.

28. Zhuang J., Tang T., Ding Y., *et al.* AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients // Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual / ed. by H. Larochelle, M. Ranzato, R. Hadsell, *et al.* 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/d9d4f495e875a2e075a1a4a6e1b9770f-Abstract.html.

29. Baydin A.G., Pearlmutter B.A., Radul A.A., Siskind J.M. Automatic Differentiation in Machine Learning: a Survey // J. Mach. Learn. Res. 2017. Vol. 18, no. 153. P. 1–43. URL: https://jmlr.org/papers/v18/17-468.html.

30. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университет // Вычислительные методы и программирование. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

31. Trindade A. Electricity Load Diagrams 2011–2014. 2015. DOI: 10.24432/C58C86. UCI Machine Learning Repository.

32. 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. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

33. Sheppy M., Beach A., Pless S. NREL RSF Measured Data 2011. Nov. 2014. Accessed: 2023-09-03 DOI: 10.25984/1845288.

34. Reiss A. PAMAP2 Physical Activity Monitoring. 2012. Accessed: 2023-09-03 DOI: 10.24432/C5NW2H. UCI Machine Learning Repository.

35. Юртин А.А. Восстановление многомерных временных рядов на основе выявления поведенческих шаблонов и применения автоэнкодеров // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 39–55. DOI: 10.14529/cmse240203.

36. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. Accessed: 2023-09-03. https://www.hydrodaten.admin.ch/.

37. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F. Elder, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

38. MeteoSwiss: Federal Office of Meteorology and Climatology. 2023. Accessed: 2023-09-03. https://www.meteoswiss.admin.ch/services-and-publications/service/open-government-data.html.

39. Weather Station Saaleaue, Max Planck Institute for Biogeochemistry, Germany. Accessed: 2023-09-03. https://www.bgc-jena.mpg.de/wetter/weather_data.html.

40. 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. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.

41. 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. Vol. 29, no. 12. P. 2744–2757. DOI: 10.1109/TKDE.2017.2750669.

42. Цымблер М.Л., Полонский В.А., Юртин А.А. Об одном методе восстановления пропущенных значений потокового временного ряда в режиме реального времени // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 4. С. 5–25. DOI: 10.14529/cmse210401.

43. Pontes F.J., F. de Amorim G. da, Balestrassi P.P., *et al.* Design of experiments and focused grid search for neural network parameter optimization // Neurocomputing. 2016. Vol. 186. P. 22–34. DOI: 10.1016/J.NEUCOM.2015.12.061.

Юртин Алексей Артемьевич, программист, Лаборатория больших данных и машинного обучения, аспирант кафедры системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240404

TOWARDS A LOSS FUNCTION FOR TRAINING NEURAL NETWORK MODELS OF TIME SERIES IMPUTATION

© 2024 A.A. Yurtin

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

E-mail: iurtinaa@susu.ru

Received: 01.09.2024

In the article, we touch upon the problem of choosing a loss function for training neural network models for imputation of missing values of multidimensional time series and introduce a novel loss function called MPDE (Mean Profile Distance Error). The MPDE function for real and reconstructed m-length subsequences is calculated as the average of the distances between all pairs of ℓ-length sliding windows of these subsequences, where $\ell \leqslant m$ and above windows have the same starting points. The distance between two windows is a modification of the MPdist (matrix profile distance) similarity measure and is defined as the weighted sum of the Euclidean and z-normalized Euclidean distances between these windows. The above weights are taken from the range [0,1] and are the hyperparameters of the loss function. When training a neural network model, MPDE allows taking into account the behavioral similarity of the compared subsequences through the presence of similar windows in them, regardless of the relative locations of these windows. Since MPDE has a high computational complexity, we implement a parallel algorithm for its calculation on a GPU to incorporate MPDE into deep learning frameworks. The algorithm is implemented using the PyTorch framework, where MPDE is represented as a sequence of automatically parallelizable operations with multidimensional tensors. Experiments over multidimensional time series from various subject domains showed that in 78% of cases state-of-the-art neural network models achieve their highest imputation accuracy (in terms of the RMSE metric) when using the proposed loss function; at the same time, the above models demonstrate imputation accuracy 40% higher than the average value achieved when using other loss functions.

Keywords: time series, imputation of missing values, neural networks, loss function, PyTorch.

FOR CITATION

Yurtin A.A. Towards a Loss Function for Training Neural Network Models of Time Series Imputation. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 4. P. 53–73. (in Russian) DOI: 10.14529/cmse240404.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Majumdar S., Laha A.K. Clustering and classification of time series using topological data analysis with applications to finance. Expert Syst. Appl. 2020. Vol. 162. P. 113868. DOI: 10.1016/j.eswa.2020.113868.
2. Lara-Benıtez P., Carranza-Garcıa M., Luna-Romera J.M., Riquelme J.C. Temporal convolutional networks applied to energy-related time series forecasting. applied sciences. 2020. Vol. 10, no. 7. P. 2322. DOI: 10.3390/app10072322.
3. Gratius N., Wang Z., Hwang M.Y., *et al.* Digital Twin Technologies for Autonomous Environmental Control and Life Support Systems. J. Aerosp. Inf. Syst. 2024. Vol. 21, no. 4. P. 332–347. DOI: 10.2514/1.I011320.
4. Zhou Z., Tang W., Li M., *et al.* A Novel Hybrid Intelligent SOPDEL Model with Comprehensive Data Preprocessing for Long-Time-Series Climate Prediction. Remote. Sens. 2023. Vol. 15, no. 7. P. 1951. DOI: 10.3390/RS15071951.
5. Kazijevs M., Samad M.D. Deep imputation of missing values in time series health data: A review with benchmarking. J. Biomed. Informatics. 2023. Vol. 144. P. 104440. DOI: 10.1016/J.JBI.2023.104440.
6. Fang C., Wang C. Time Series Data Imputation: A Survey on Deep Learning Approaches. CoRR. 2020. Vol. abs/2011.11347. arXiv: 2011.11347. URL: https://arxiv.org/abs/2011.11347.
7. Cao W., Wang D., Li J., *et al.* BRITS: Bidirectional Recurrent Imputation for Time Series. Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada / ed. by S. Bengio, H.M. Wallach, H. Larochelle, *et al.* 2018. P. 6776–6786. URL: https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html.
8. Yoon J., Zame W.R., Schaar M. van der Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks. IEEE Trans. Biomed. Eng. 2019. Vol. 66, no. 5. P. 1477–1490. DOI: 10.1109/TBME.2018.2874712.
9. Fortuin V., Baranchuk D., Rätsch G., Mandt S. GP-VAE: Deep Probabilistic Time Series Imputation. The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Vol. 108 / ed. by S. Chiappa, R. Calandra. PMLR, 2020. P. 1651–1661. Proceedings of Machine Learning Research. URL: http://proceedings.mlr.press/v108/fortuin20a.html.

10. Du W., Côté D., Liu Y. SAITS: Self-attention-based imputation for time series. Expert Syst. Appl. 2023. Vol. 219. P. 119619. DOI: 10.1016/J.ESWA.2023.119619.

11. Oh E., Kim T., Ji Y., Khyalia S. STING: Self-attention based Time-series Imputation Networks using GAN. 2021. DOI: 10.1109/ICDM51629.2021.00155.

12. Kaya M., Bilge H.S. Deep Metric Learning: A Survey. Symmetry. 2019. Vol. 11, no. 9. P. 1066. DOI: 10.3390/SYM11091066.

13. Wang Q., Ma Y., Zhao K., Tian Y. A comprehensive survey of loss functions in machine learning. Annals of Data Science. 2022. Vol. 9. P. 187–212. DOI: 10.1007/s40745-020-00253-5.

14. Ciampiconi L., Elwood A., Leonardi M., *et al.* A survey and taxonomy of loss functions in machine learning. CoRR. 2023. Vol. abs/2301.05579. DOI: 10.48550/ARXIV.2301.05579. arXiv: 2301.05579.

15. Netrapalli P. Stochastic gradient descent and its variants in machine learning. Journal of the Indian Institute of Science. 2019. Vol. 99, no. 2. P. 201–213. DOI: 10.1007/s41745-019-0098-4.

16. Xu J., Ren X., Lin J., Sun X. Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018 / ed. by E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii. Association for Computational Linguistics, 2018. P. 3940–3949. DOI: 10.18653/V1/D18-1428.

17. Chang Y., Wang X., Wang J., *et al.* A Survey on Evaluation of Large Language Models. ACM Trans. Intell. Syst. Technol. 2024. Vol. 15, no. 39. P. 1–45. DOI: 10.1145/3641289.

18. Chen W., Huang H., Peng S., *et al.* YOLO-face: a real-time face detector. Vis. Comput. 2021. Vol. 37, no. 4. P. 805–813. DOI: 10.1007/S00371-020-01831-7.

19. Paszke A., Gross S., Massa F., *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada / ed. by H.M. Wallach, H. Larochelle, A. Beygelzimer, *et al.* 2019. P. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

20. Jadon A., Patil A., Jadon S. A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting. International Conference on Data Management, Analytics & Innovation. Springer. 2024. P. 117–147. DOI: 10.1007/978-981-97-3245-6_9.

21. Qi J., Du J., Siniscalchi S.M., *et al.* On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression. IEEE Signal Process. Lett. 2020. Vol. 27. P. 1485–1489. DOI: 10.1109/LSP.2020.3016837.

22. Chen X., Liu W., Mao X., Yang Z. Distributed High-dimensional Regression Under a Quantile Loss Function. J. Mach. Learn. Res. 2020. Vol. 21, no. 182. P. 1–43. URL: http://jmlr.org/papers/v21/20-297.html.

23. Saleh R.A., Saleh A.K.M.E. Statistical Properties of the log-cosh Loss Function Used in Machine Learning. CoRR. 2022. Vol. abs/2208.04564. DOI: 10.48550/ARXIV.2208.04564. arXiv: 2208.04564.

24. Sun Q., Zhou W.-X., Fan J. Adaptive Huber regression. Journal of the American Statistical Association. 2020. Vol. 115, no. 529. P. 254–265. DOI: 10.1080/01621459.2018.1543124.

25. Cuturi M., Blondel M. Soft-DTW: a Differentiable Loss Function for Time-Series. Proceedings of the 34th International Conference on Machine Learning. Vol. 70 / ed. by D. Precup, Y.W. Teh. PMLR, June 2017. P. 894–903. Proceedings of Machine Learning Research. URL: https://proceedings.mlr.press/v70/cuturi17a.html.

26. Berndt D.J., Clifford J. Using Dynamic Time Warping to find patterns in time series. KDD Workshop. 1994. P. 359–370. URL: https://cdn.aaai.org/Workshops/1994/WS-94-03/WS94-03-031.pdf.

27. Gharghabi S., Imani S., Bagnall A.J., *et al.* Matrix Profile XII: MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios. IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018. IEEE Computer Society, 2018. P. 965–970. DOI: 10.1109/ICDM.2018.00119.

28. Zhuang J., Tang T., Ding Y., *et al.* AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual / ed. by H. Larochelle, M. Ranzato, R. Hadsell, *et al.* 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/d9d4f495e875a2e075a1a4a6e1b9770f-Abstract.html.

29. Baydin A.G., Pearlmutter B.A., Radul A.A., Siskind J.M. Automatic Differentiation in Machine Learning: a Survey. J. Mach. Learn. Res. 2017. Vol. 18, no. 153. P. 1–43. URL: https://jmlr.org/papers/v18/17-468.html.

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

31. Trindade A. Electricity Load Diagrams 2011–2014. 2015. DOI: 10.24432/C58C86. UCI Machine Learning Repository.

32. 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. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

33. Sheppy M., Beach A., Pless S. NREL RSF Measured Data 2011. Nov. 2014. Accessed: 2023-09-03 DOI: 10.25984/1845288.

34. Reiss A. PAMAP2 Physical Activity Monitoring. 2012. Accessed: 2023-09-03 DOI: 10.24432/C5NW2H. UCI Machine Learning Repository.

35. Yurtin A.A. Imputation of Multivariate Time Series Based on the Behavioral Patterns and Autoencoders. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 39–55. DOI: 10.14529/cmse240203.

36. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. Accessed: 2023-09-03. https://www.hydrodaten.admin.ch/.

37. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F. Elder, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

38. MeteoSwiss: Federal Office of Meteorology and Climatology. 2023. Accessed: 2023-09-03. https://www.meteoswiss.admin.ch/services-and-publications/service/open-government-data.html.

39. Weather Station Saaleaue, Max Planck Institute for Biogeochemistry, Germany. Accessed: 2023-09-03. https://www.bgc-jena.mpg.de/wetter/weather_data.html.

40. 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. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.

41. 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. Vol. 29, no. 12. P. 2744–2757. DOI: 10.1109/TKDE.2017.2750669.

42. Zymbler M.L., Polonsky V.A., Yurtin A.A. On One Method of Imputation Missing Values of a Streaming Time Series in Real Time. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 4. P. 5–25. DOI: 10.14529/cmse210401.

43. Pontes F.J., F. de Amorim G. da, Balestrassi P.P., *et al.* Design of experiments and focused grid search for neural network parameter optimization. Neurocomputing. 2016. Vol. 186. P. 22–34. DOI: 10.1016/J.NEUCOM.2015.12.061.