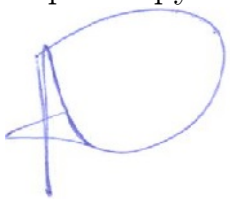
Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(национальный исследовательский университет)»

На правах рукописи



ЮРТИН Алексей Артемьевич

НЕЙРОСЕТЕВЫЕ МЕТОДЫ ВОССТАНОВЛЕНИЯ ПОТОКОВЫХ ДАННЫХ

Специальность 2.3.5. Математическое и программное обеспечение
вычислительных систем, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:

ЦЫМБЛЕР Михаил Леонидович,
доктор физ.-мат. наук, доцент

Челябинск — 2025

Оглавление

Введение **5**

1 Современные методы восстановления потоковых данных, представленных в форме временных рядов **22**

1.1 Формальные определения и обозначения 22

1.1.1 Временной ряд и подпоследовательность 23

1.1.2 Поведенческие шаблоны временного ряда 24

1.2 Жизненный цикл нейросетевых моделей 26

1.3 Таксономия методов восстановления потоковых данных, представленных в форме временных рядов 32

1.3.1 Аналитические методы 34

1.3.2 Нейросетевые методы 40

1.4 Обзор близких работ 44

1.4.1 Нейросетевые методы восстановления потоковых данных 44

1.4.2 Функции потерь нейросетевых моделей восстановления потоковых данных 47

1.4.3 Методы предварительной оценки нейросетевых моделей восстановления потоковых данных 49

1.5 Выводы по главе 1 51

2 Нейросетевые методы и модели восстановления потоковых данных, представленных в форме временных рядов **53**

2.1 Восстановление в режиме онлайн 53

2.1.1 Нейросетевой метод SANNI 54

2.1.2 Компоненты метода 55

2.1.3 Обучение нейросетевых моделей 63

2.1.4 Применение метода 66

2.2 Восстановление в режиме офлайн 67

2.2.1 Нейросетевой метод SAETI 68
2.2.2 Компоненты метода 69
2.2.3 Обучение нейросетевых моделей 76
2.2.4 Применение метода 80
2.3 Выводы по главе 2 80

3 Оценка точности нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов 82

3.1 Функция потерь для обучения нейросетевых моделей восстановления 82
3.1.1 Формальное определение функции потерь MPDE . . 83
3.1.2 Параллельный алгоритм вычисления 97
3.2 Метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления 101
3.2.1 Нейросетевой метод tsGAP 101
3.2.2 Компоненты метода 105
3.2.3 Обучение нейросетевых моделей 113
3.3 Выводы по главе 3 117

4 Вычислительные эксперименты 120

4.1 Наборы данных и сценарии формирования пропусков 120
4.2 Нейросетевой метод SANNI 124
4.2.1 Описание экспериментов 124
4.2.2 Анализ результатов 125
4.3 Нейросетевой метод SAETI 131
4.3.1 Описание экспериментов 131
4.3.2 Анализ результатов 132
4.4 Функция потерь MPDE 136
4.4.1 Описание экспериментов 136
4.4.2 Анализ результатов 138

4.5 Метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления tsGAP 149
4.5.1 Описание экспериментов 150
4.5.2 Анализ результатов 154
4.6 Выводы по главе 4 156

Заключение **159**

Список литературы **164**

Приложение А. Результаты вычислительных экспериментов для метода SANNI **188**

Приложение В. Результаты вычислительных экспериментов для функции потерь MPDE **190**

Приложение С. Структура входных данных метода tsGAP **192**

Введение

Актуальность темы исследования

Актуальность темы диссертационного исследования обусловлена следующими основными факторами:

1) широкое распространение потоковых данных в прикладных задачах и высокая практическая значимость задачи их восстановления;

2) интенсивные научные исследования, направленные на повышение точности методов глубокого обучения, применяемых для восстановления потоковых данных;

3) актуальность задачи повышения эффективности жизненного цикла нейросетевых моделей восстановления потоковых данных.

Широкое распространение потоковых данных. В современном информационном обществе типичной является ситуация, когда данные поступают в непрерывном потоке. В широком спектре предметных областей сенсоры, журналы событий, сетевые источники и другие средства мониторинга генерируют постоянный поток информации. Одной из наиболее распространенных форм представления таких данных являются *временные ряды* [50, 145]. *Временной ряд* представляет собой упорядоченную последовательность наблюдений, фиксируемых через равные интервалы времени и отражающих динамику исследуемых процессов [65, 136]. Временной ряд, содержащий наблюдения одного показателя во времени, называется *одномерным*. Современные временные ряды часто являются *многомерными*, представляя собой несколько синхронизированных по времени одномерных рядов, каждый из которых описывает отдельный аспект наблюдаемого явления.

В последнее десятилетие объем собираемых данных, относящихся к потоковым данных, значительно возрос, затрудняя их ручной анализ [16].

Для решения этой проблемы все чаще применяются методы интеллектуального анализа временных рядов, основанные на методах машинного обучения и статистического моделирования, и извлекающие полезную информацию из больших объемов данных [45]. Подобные методы успешно используются для решения задач в различных предметных областях: здравоохранение [19], суперкомпьютерные системы [2], моделирование климата [166], Интернета вещей [70, 80], энергетика [83], цифровые двойники [5], интеллектуальное управление сложными системами [7, 55], финансовое прогнозирование [4], транспорт [165], кибербезопасность [82], юриспруденция [3], экология [156] и др.

Важность задачи восстановления пропусков. Восстановление пропущенных значений является одной из ключевых задач интеллектуального анализа потоковых данных [12, 47, 76, 140]. Пропуски могут возникать по разным причинам [108, 151]: технические сбои, ошибки сбора данных, влияние человеческого фактора и др. Наличие пропусков существенно снижает качество аналитических выводов, приводя к искажению результатов работы моделей и ошибочным интерпретациям данных [32, 54]. Отсутствие корректной обработки пропусков ограничивает возможности выявления закономерностей, прогнозирования и принятия обоснованных решений [54, 146]. Современные методы восстановления пропусков стремятся минимизировать искажения, вызванные неполнотой данных, обеспечивая при этом сохранение временной структуры и статистических характеристик.

Во многих прикладных областях потоковые данные, представленные в форме временных рядов, характеризуются наличием устойчиво повторяющихся паттернов, соответствующих активностям или поведенческим сценариям исследуемой системы [87, 103]. Под *активностью* в данном контексте понимается одно из возможных состояний анализируемого объекта, характеризующееся специфическим типом поведения, сохраняющимся в течение определенного временного интервала [91]. Поведение в рамках одной активности, как правило, обладает внутренней структурной и статистической однородностью, не демонстрирует резких переходов, выражен-

ной хаотичности или существенных отклонений. Указанная однородность может проявляться в виде повторяемости характерных паттернов, относительной стабильности амплитудно-частотных характеристик или устойчивой формы сигнала. В рамках таких активностей нередко наблюдаются участки временного ряда, демонстрирующие устойчивое, воспроизводимое поведение. Подобные участки можно интерпретировать как *поведенческие шаблоны*. Под *поведенческим шаблоном* (snippet) можно понимать наиболее репрезентативный и характерный паттерн временного ряда, который отражает динамику исследуемого объекта в рамках определенной активности [67]. В контексте восстановления потоковые данные, представленных в форме временных рядов, важно обеспечивать восстановление, сохраняющее поведенческие особенности анализируемых данных. Для повышения точности восстановления потенциально могут использоваться поведенческие шаблоны, которые позволяют воспроизводить характерные паттерны динамики объекта в пропущенных интервалах и сохранять структурные и статистические особенности данных.

В зависимости от режима доступа к данным и предполагаемой области применения существующие методы восстановления можно разделить на офлайн и онлайн [41]. *Офлайн методы* предполагают наличие полного доступа ко всей последовательности потоковых данных, включая данные как до, так и после пропущенных значений. Такие методы, как правило, обладают большей вычислительной сложностью, но обеспечивают более точное восстановление за счет использования всей доступной информации, окружающей пропуск. Подобные методы особенно эффективны при пакетной обработке архивных данных и в случаях, когда задержка в обработке допустима. *Онлайн методы*, в отличие от офлайн, выполняют восстановление в условиях отсутствия информации о значениях после пропуска и ориентированы на непрерывную обработку данных по мере их поступления. Такие методы могут применяться в системах реального времени, которые накладывают дополнительные ограничения на время отклика и требуют, чтобы задержки обработки данных не превышали допустимый порог [89].

Онлайн методы имеют принципиальное значение для систем непрерывного мониторинга, где высокая скорость восстановления данных необходима для минимизации задержек и предотвращения принятия неоптимальных или ошибочных решений [12].

Задачи восстановления и прогноза потоковых данных являются смежными, однако, несмотря на методологическую схожесть, они имеют принципиальные различия [15]. Восстановление направлено на заполнение пропусков в уже наблюдаемых данных и, даже в режиме онлайн, заключается в восстановлении значения в текущей временной точке на основе доступной истории. Прогнозирование, напротив, предполагает предсказание будущих значений ряда, которые еще не наблюдались, и использует исключительно доступные данные до текущего момента. При поступлении новых наблюдений формируется новый прогноз на основе актуальных полных данных, тогда как нейросетевая модель восстановления продолжает работу с частично доступными или ранее восстановленными значениями, рассматривая их как входные данные для последующих шагов восстановления. В отличие от прогноза, который всегда опирается на достоверные наблюдения, восстановление функционирует в условиях неполноты данных и вынуждено учитывать собственные предыдущие оценки, накладывая тем самым дополнительные требования к устойчивости модели и ограничению накопления ошибок в процессе работы.

Развитие нейросетевых методов восстановления данных. В условиях значительного роста объема и усложнения структуры потоковых данных классические статистические методы восстановления часто оказываются недостаточно точными при моделировании сложных нелинейных и многомерных зависимостей. В ответ на эти вызовы активно развиваются нейросетевые подходы к восстановлению пропущенных значений. Архитектуры на основе рекуррентных нейронных сетей, автоэнкодеров и трансформеров демонстрируют высокую способность моделировать временные и пространственные зависимости и адаптироваться к разнообразным шаблонам пропусков [72]. Несмотря на высокую гибкость и адаптивность нейро-

сетевых моделей, их практическое применение сопряжено с рядом методологических и вычислительных сложностей. Одной из ключевых задач при разработке нейросетевых моделей является выбор адекватной функции потерь, учитывающей специфику задачи и характер пропусков. В контексте восстановления потоковых данных, представленных в форме временных рядов, важно, чтобы функция потерь при сравнении исходных и восстановленных данных учитывала особенности рядов и их поведенческое сходство. Если эти критерии не будут отражены, нейросетевая модель, обученная с использованием такой функции потерь, будет генерировать значения, не соответствующие реальному поведению объекта, что снижает точность восстановления, корректность аналитических выводов и практическую ценность восстановленных данных.

Важность задачи прогнозирования ошибки и времени обучения нейросетевых моделей. Не менее важной проблемой является выбор архитектуры нейросетевой модели. Архитектура нейросетевой модели существенно влияет на эффективность восстановления потоковых данных. Современные нейросетевые модели отличаются высокой вариативностью по числу параметров, глубине, механизму обработки временной информации и методам регуляризации, что делает выбор наиболее точного решения нетривиальной задачей [43]. В условиях, когда перебор всех возможных конфигураций становится вычислительно затратным, возникает необходимость в разработке методов предварительного прогнозирования точности нейросетевых моделей. Подобные методы позволяют на основе характеристик данных, структуры пропусков и мета-признаков нейросетевой модели прогнозировать ее ожидаемую точность восстановления, тем самым существенно снижая потребность в ресурсоемких экспериментах.

Жизненный цикл нейросетевых моделей восстановления потоковых данных. В современных системах машинного обучения восстановление потоковых данных, представленных в форме временных рядов, можно рассматривать как *жизненный цикл* нейросетевых моделей — замкнутую последовательность взаимосвязанных этапов, каждый из которых

выполняет специализированную функцию и обеспечивает корректность последующей обработки данных [18]. В рамках такого подхода ключевые этапы (сбор и предварительная обработка данных, выбор, обучение и применение нейросетевой модели) рассматриваются не как изолированные задачи, а как часть единого процесса, обеспечивающего максимизацию точности восстановления. Ошибки на любом из этапов могут привести к каскадному ухудшению качества.

Учитывая изложенные выше факторы, можно заключить, что задача проведения исследований, направленных на разработку новых методов и повышение точности существующих подходов к восстановлению потоковых данных, представленных в форме временных рядов, является актуальной и находится в фокусе интенсивных научных исследований и практических разработок.

Объект и предмет исследования

Объектом исследования являются системы обработки и анализа потоковых данных. *Предметом исследования* выступают модели, методы и алгоритмы восстановления пропущенных значений в потоковых данных.

Степень разработанности темы

Ранними подходами к предварительной обработке пропущенных значений можно считать либо исключение неполных наблюдений из анализа, либо использование методов, не учитывающих временную зависимость данных [57]. Например, в 1963 г. в своей работе Кособуд (Kosobud) [78] указывает, что восстановление пропущенных значений может основываться на использовании пар переменных с высокой корреляцией, позволяя оценивать пропуски одной переменной через известные значения другой. В дальнейшем Афифи (Afifi) и Элашофф (Elashoff) [13] представили систематический обзор методов восстановления пропущенных данных, включающий классический метод подстановки средних значений, метод максимального

правдоподобия и байесовские методы. Однако результаты их исследований показали, что ни один из подходов не обладает универсальной эффективностью, поскольку точность метода зависит от структуры данных, характера распределения признаков и степени коррелированности между переменными.

Существенный прогресс в развитии методов восстановления пропущенных значений произошел в тот момент, когда при обработке данных начали учитывать временную зависимость последовательности наблюдений. В работе, посвященной выявлению трендов в данных качества воды, Леттенмайер (Lettenmaier) предложил использовать модель Маркова первого порядка для предварительной обработки пропущенных значений [85]. Пропущенные значения синтезируются на основе сезонных средних, стандартного отклонения и автокорреляции с шагом в одно значение. В результате работы модели формируется псевдоисторическая последовательность, воспроизводящая основные статистические характеристики исходного временного ряда и обеспечивающая согласованность восстановленных значений с динамикой наблюдаемых данных. Для применения ARIMA (AutoRegressive Integrated Moving Average) в задачах восстановления временных рядов в 1983 г. Харви (Harvey) и Пирс (Pierce) предложили метод оценки параметров модели ARIMA на основе максимального правдоподобия. Однако предложенный подход требовал наличия непрерывного фрагмента наблюдений в начале или в конце временного ряда, что ограничивало его применимость при большом количестве пропусков. В 1989 г. Бэк (Beck) [24] продемонстрировал эффективность фильтра Калмана на примере предварительной обработки данных социальных опросов об уровне одобрения деятельности администрации Рейгана, в которых 40% месячных наблюдений отсутствовало.

Одновременно с этим происходит развитие рекуррентных нейронных сетей (recurrent neural network, RNN). В 1989 г. Вильямс (Williams) и Зипсер (Zipser) описали алгоритм обучения рекуррентных нейросетей, основанный на градиентном спуске [144]. Авторы указывали на практическую

ценность RNN для анализа временных рядов благодаря их способности моделировать последовательные зависимости между наблюдениями. Однако классические RNN сталкивались с проблемой затухающих градиентов, которая приводила к потере информации на длительных временных интервалах. В 1997 г. Хохрейтер (Hochreiter) и Шмидхубер (Schmidhuber) [62] предложили архитектуру LSTM (Long Short-Term Memory), которая предназначена для решения этой проблемы и обеспечивает эффективное моделирование долгосрочных зависимостей в последовательных данных. Благодаря наличию дополнительных параметров LSTM-блоки способны оценивать важность информации на каждом временном шагу, избирательно обновлять внутреннее состояние и сохранять наиболее значимые данные для дальнейшей обработки. Увеличение числа параметров приводило к высоким вычислительным затратам, что осложняло масштабирование моделей на основе LSTM в задачах обработки временных рядов. Частично эту проблему удалось решить в 2014 г. Чо (Cho) и др. [35] в своей работе предложили новый рекуррентный блок GRU (Gated Recurrent Unit), который в отличие от LSTM обладал меньшим числом параметров и сохранял способность моделировать длительные зависимости в данных.

На основе рассмотренных выше архитектур и их ключевых элементов впоследствии были разработаны различные методы восстановления пропущенных значений, использующие нейросетевые модели. Например, в 2018 г. Као (Cao) и др. предложили нейросетевую модель BRITS (Bidirectional Recurrent Imputation for Time Series) [30], представляющую собой двунаправленную рекуррентную нейронную сеть, последовательно восстанавливающую пропуски во временном ряду. В качестве другого примера можно привести нейросетевую модель GP-VAE (Gaussian Process Variational Autoencoder), предложенную Фортьен (Fortuin) и др. в 2020 г. [49]. Нейросетевая модель использует сверточные нейронные сети для формирования скрытого представления подпоследовательности временного ряда и гауссовский процесс для моделирования зависимостей между точками данных в получившемся пространстве.

В 2023 г. Ду (Du) и др. предложили нейросетевую модель SAITS (Self-Attention-based Imputation for Time Series) [44], которая использует механизм внимания [132, 141] для восстановления пропущенных значений во временных рядах. Применение механизма внимания позволяет учитывать сложные временные зависимости и связи между различными точками многомерных временных рядов, улучшая качество восстановления пропусков по сравнению с рекуррентными нейросетевыми моделями.

Несмотря на высокую точность восстановления, указанные методы, как правило, не учитывают шаблоны. Включение поведенческих шаблонов в процесс обработки данных могло бы повысить точность восстановления. Шаблоны могли бы улучшать адаптацию нейросетевой модели к новым данным, повышая интерпретируемость результатов и предоставляя пример ожидаемого поведения.

Другим существенным ограничением применимости нейросетевых методов является наличие большого числа гиперпараметров и конструктивных элементов архитектуры, таких как количество слоев, размер скрытого состояния, тип функций активации, способы регуляризации и др. Например, в 2020 г. Донг (Dong) и др. в своей работе [43], посвященной методам поиска оптимальных нейросетевых моделей, рассматривают около 16 000 уникальных архитектур. Исследование столь большого числа архитектур для каждой отдельной задачи восстановления пропущенных значений затруднительно. Вследствие этого выбор архитектуры, обладающей высокой точностью и малыми вычислительными затратами на обработку данных, представляет собой сложную задачу, требующую учета структуры данных, характеристик пропусков и особенностей нейросетевых слоев. В связи с этим естественным шагом становится развитие Neural Architecture Search (NAS) [43] — методов автоматизированного проектирования архитектуры нейросети. Методы NAS все чаще применяются в рамках AutoML (Automated Machine Learning) [59] для автоматического подбора архитектуры нейросети и в MLOps (Machine Learning Operations) [126] на различных этапах жизненного цикла нейросетевых моделей.

Исходя из рассмотренного выше, можно заключить, что на сегодняшний день задача разработки эффективных методов и алгоритмов восстановления пропущенных значений потоковых данных, представленных в форме временных рядов, остается в фокусе интенсивных научных исследований и практических разработок.

Цель и задачи исследования

Цель данной работы состояла в разработке новых и повышении эффективности существующих методов восстановления потоковых данных, представленных в форме многомерных временных рядов, на основе совместного использования нейросетевых моделей и поведенческих шаблонов. Данная цель предполагает решение следующих *задач*.

1. Разработать и исследовать комплекс нейросетевых моделей, методов и алгоритмов, предназначенных для восстановления потоковых данных, представленных в форме временных рядов, и включающий в себя следующие компоненты:
   - нейросетевые методы восстановления многомерных временных рядов в офлайн и онлайн режимах;
   - функцию потерь для обучения нейросетевых моделей восстановления временных рядов, учитывающую поведенческий контекст;
   - метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления временных рядов.
2. Провести вычислительные эксперименты с синтетическими и реальными временными рядами, исследующие эффективность предложенных нейросетевых моделей, методов и алгоритмов.

Научная новизна

Научная новизна проведенного диссертационного исследования заключается в следующем.

1. Разработаны новые нейросетевые методы восстановления потоковых данных, представленных в форме временных рядов, в режимах офлайн и онлайн, впервые использующие концепцию поведенческих шаблонов, которая обеспечивает повышение точности восстановления.

2. Предложена новая функция потерь, обеспечивающая повышение точности восстановления потоковых данных, представленных в форме временных рядов, нейросетевыми моделями за счет учета поведенческого сходства подпоследовательностей во время обучения. Разработан эффективный параллельный алгоритм, позволяющий интегрировать предложенную функцию потерь в современные фреймворки глубокого обучения.

3. Разработан новый метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, основанный на представлении обучаемых нейросетевых моделей в виде направленного ациклического графа и обеспечивающий возможность предварительной оценки точности нейросетевых моделей без необходимости их обучения.

Теоретическая и практическая значимость работы

Теоретическая ценность диссертационной работы состоит в следующем. В разработанных методах восстановления потоковых данных предложены оригинальные архитектуры нейросетевых моделей, позволяющие использовать в качестве дополнительного источника информации шаблоны, извлекаемые из исходных данных. Разработанная функция потерь нейросетевых моделей восстановления потоковых данных, в отличие от аналогов, обеспечивающих геометрическую интерпретацию точек, позволяет учитывать во время обучения поведенческую схожесть исходных данных. Предложена нейросетевая модель, позволяющая прогнозировать ошибку и время обучения моделей восстановления потоковых данных, представленных в форме многомерных временных рядов, без необходимости их обучения.

Практическая ценность диссертационной работы заключается в том, что разработанные нейросетевые модели, методы и алгоритмы могут быть использованы для решения задач, связанных с восстановлением временных рядов в интеллектуальных системах обработки потоковых данных. Использование предложенных методов и моделей в качестве инструментальных средств разработки цифровых продуктов позволит автоматизировать подготовку и обработку потоковых данных, содержащих пропуски, минимизируя участие человека за счет автоматизированного восстановления. Полученные методы могут быть использованы в системах AutoML, MLOps и иных интеллектуальных системах машинного обучения в качестве инструмента для повышения устойчивости и точности анализа потоковых данных, представленных в форме временных рядов.

Методология и методы исследования

Методологической основой диссертационной работы являются методы глубокого обучения, интеллектуального анализа данных и теория временных рядов. Для программной реализации разработанных подходов применялись методы объектно-ориентированного проектирования и разработки нейросетевых моделей с использованием фреймворка PyTorch.

Положения, выносимые на защиту

На защиту выносятся следующие новые научные результаты.

1. Разработан нейросетевой метод SANNI для восстановления потоковых данных, представленных в форме временных рядов, в режиме онлайн, который для повышения точности восстановления использует поведенческие шаблоны, извлекаемые из репрезентативного фрагмента исходных данных.

2. Разработан нейросетевой метод SAETI для восстановления потоковых данных, представленных в форме временных рядов, в режиме офлайн,

который для повышения точности восстановления использует поведенческие шаблоны, извлекаемые из репрезентативного фрагмента исходных данных.

3. Разработана функция потерь MPDE, предназначенная для обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Предложенная функция представляет собой меру поведенческой схожести временных рядов, в отличие от традиционных функций потерь, основанных на геометрической интерпретации ошибок. Реализован параллельный алгоритм вычисления данной функции, обеспечивающий возможность ее использования в современных фреймворках глубокого обучения.

4. Разработан нейросетевой метод tsGAP для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, который для повышения точности прогноза использует графовое представление нейросетевых моделей восстановления.

Степень достоверности результатов

Эффективность разработанных методов и алгоритмов подтверждена результатами вычислительных экспериментов, проведенных на реальных и синтетических данных в соответствии с общепринятыми стандартами. Репродуцируемость полученных результатов обеспечивается размещением исходного кода, реализующего предложенные нейросетевые модели, методы и алгоритмы, в свободно доступных репозиториях в сети Интернет.

Соответствие диссертации паспорту научной специальности

Тематика, содержание и полученные результаты диссертационного исследования соответствуют следующему направлению исследований паспорта специальности 2.3.5. «Математическое и программное обеспечение вы-

числительных систем, комплексов и компьютерных сетей»: п. 4. Интеллектуальные системы машинного обучения, управления базами данных и знаний, инструментальные средства разработки цифровых продуктов.

Апробация результатов исследования

Основные положения диссертационной работы, разработанные методы, алгоритмы и результаты вычислительных экспериментов докладывались на *пяти научных конференциях*:

— ПаВТ'2025: Всероссийская научная конференция с международным участием «Параллельные вычислительные технологии 2025», 8–10 апреля 2025 г., Москва;

— DAMDID'2024: International Conference on Data Analytics and Management in Data Intensive Domains, 23–25 October 2024, Nizhny Novgorod, Russia;

— ПаВТ'2024: Всероссийская научная конференция с международным участием «Параллельные вычислительные технологии 2024», 2–4 апреля 2024 г., Челябинск *(диплом I степени в конкурсе докладов молодых ученых)*;

— ЦИСП'2023: Всероссийская научная конференция с международным участием «Цифровая индустрия: состояние и перспективы развития 2023», 21–23 ноября 2023 г., Челябинск;

— ПаВТ'2023: Всероссийская научная конференция с международным участием «Параллельные вычислительные технологии 2023», 28–30 марта 2023 г., Санкт-Петербург *(диплом II степени в конкурсе докладов молодых ученых)*.

Публикации соискателя по теме диссертации

Основные результаты диссертации опубликованы в *пяти научных работах* в журналах категорий К1 и К2 Перечня ВАК, в том числе 4 статьи опубликованы в журналах, входящих в Ядро РИНЦ (где одна из статей опубликована в журнале, который также входит в квартиль Q2 библиографической базы данных Scopus).

1. Yurtin, A. SANNI: Online Imputation of Missing Values in Multivariate Time Series Based on Deep Learning and Behavioral Patterns / A. Yurtin, M. Zymbler // Lobachevskii Journal of Mathematics. – 2024. – Vol. 45, no. 11. – P. 5948-–5966. DOI: 10.1134/S1995080224606854. WOS: 001446922600003. (Ядро РИНЦ, Перечень ВАК К1, Scopus Q2).

2. Юртин А.А. Метод прогнозирования ошибки времени обучения нейросетевых моделей восстановления многомерных временных рядов // Проблемы информатики. – 2025. – № 3. – С. 78–95. DOI: 10.24412/2073-0667-2025-3-72-95. (Перечень ВАК К2).

3. Юртин А.А. Об одной функции потерь для обучения нейросетевых моделей восстановления временных рядов / А.А. Юртин // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. – 2024. – Т. 13, № 4. – С. 53–73. DOI: 10.14529/cmse240404. (Ядро РИНЦ, Перечень ВАК К2).

4. Юртин А.А. Восстановление многомерных временных рядов на основе выявления поведенческих шаблонов и применения автоэнкодеров / А.А. Юртин // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. – 2024. – Т. 13, № 2. – С. 39–55. DOI: 10.14529/cmse240203. (Ядро РИНЦ, Перечень ВАК К2).

5. Цымблер М.Л., Восстановление пропущенных значений временного ряда на основе совместного применения аналитических алгоритмов и нейронных сетей / М.Л. Цымблер, А.А. Юртин // Вычислительные ме-

тоды и программирование. – 2023. – Т. 24, № 3. – С. 243–259. DOI: 10.26089/NumMet.v24r318. (Ядро РИНЦ, Перечень ВАК К1).

В статьях [1,5] научному руководителю М.Л. Цымблеру принадлежит постановка задачи, А.А. Юртину — все полученные результаты.

Структура и объем работы

Диссертация состоит из введения, четырех глав, заключения, списка литературы и трех приложений. Объем диссертации составляет 193 страницы, объем списка литературы — 168 наименований.

Содержание работы

В первой главе, «Современные методы восстановления потоковых данных, представленных в форме временных рядов», дается обзор современных методов и алгоритмов восстановления потоковых данных, представленных в форме временных рядов. Рассматриваются основные этапы жизненного цикла нейросетевых моделей восстановления. Проведен обзор публикаций, наиболее близких к тематике диссертационной работы, в котором рассматриваются нейросетевые методы восстановления, функции потерь нейросетевых моделей восстановления, а также существующие подходы к предварительной оценке точности нейросетевых моделей восстановления потоковых данных.

Во второй главе, «Нейросетевые методы и модели восстановления потоковых данных, представленных в форме временных рядов», представлены новые методы восстановления потоковых данных, представленных в форме временных рядов, в режимах онлайн и офлайн, получившие названия SANNI (Snippet and Neural Network based Imputation) и SAETI (Snippet based Autoencoder for Timeseries Imputation). Подробно описаны основные этапы обработки входных данных, архитектура нейросетевых моделей и методы их обучения.

Третья глава, «Оценка точности нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов», посвящена новым методам анализа нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. В главе предложена новая функция потерь MPDE (Mean Profile Distance Error), повышающая точность нейросетевых моделей восстановления за счет учета поведенческих особенностей подпоследовательностей в процессе обучения. Описан новый метод tsGAP (time-series Graph-Attention Perfomance Prediction model), предназначенный для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов.

Четвертая глава, «Вычислительные эксперименты», посвящена вычислительным экспериментам, исследующим эффективность предложенных методов, нейросетевых моделей и алгоритмов. В главе резюмированы наборы данных, которые использовались в экспериментах, их категории, сценарии формирования пропусков и метрики для оценки качества восстановления потоковых данных, представленных в форме временных рядов. Глава содержит сравнительный анализ предложенных методов с передовыми аналогами.

Заключение подводит итоги диссертационной работы и содержит описание ключевых отличий данного исследования от ранее выполненных родственных работ других авторов, рекомендации по использованию полученных результатов, а также направления дальнейших исследований.

Приложения A, B и C вынесены результаты вычислительных экспериментов, исследующих точность метода SANNI, функции потерь MPDE и описание структуры входных данных модели tsGAP соответственно.

Глава 1. Современные методы восстановления потоковых данных, представленных в форме временных рядов

В данной главе рассматривается проблема пропусков в потоковых данных и методы их восстановления. Приводится описание жизненного цикла нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Определяются ключевые этапы разработки нейросетевых моделей восстановления и анализируются инструментальные средства, влияющие на итоговую точность восстановления. Представлен обзор основных подходов к восстановлению пропущенных значений, включающий аналитические и нейросетевые методы. Рассмотрены алгоритмы и методы, входящие в состав каждой из указанных групп. Вводятся формальные определения и обозначения, которые будут использоваться в последующих главах. Проведен анализ научных публикаций, посвященных нейросетевым методам восстановления потоковых данных, представленных в форме временных рядов, методам повышения точности восстановления и оценки качества, наибольшим образом соответствующих тематике диссертационного исследования.

1.1. Формальные определения и обозначения

При выполнении научных исследований, связанных с обработкой потоковых данных, принято их представлять в форме временных рядов [50, 145]. В данном разделе вводятся формальные определения и нотация для обозначения используемых в последующих главах понятий в соответствии с работой [67].

1.1.1. Временной ряд и подпоследовательность

Одномерный временной ряд представляет собой хронологически упорядоченную последовательность вещественных значений:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}. \tag{1.1}$$

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

Подпоследовательность (subsequence) $T_{i,\,m}$ одномерного временного ряда T представляет собой непрерывный промежуток из m элементов ряда, начиная с i го элемента:

$$T_{i,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 1 \leqslant i \ll n - m + 1, \quad 3 \leqslant m \leqslant n. \tag{1.2}$$

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначим как S_T^m. Мощность указанного множества обозначим как N, $N = |S_T^m| = n - m + 1$. Рассмотрим подпоследовательность $T_{i,\,m}$ как самостоятельный временной ряд. Тогда его подпоследовательность, имеющую заданную длину ℓ (где $\ell < m$), будем называть *окном*. Мощность множества окон заданной подпоследовательности обозначим соответственно как c, т.е. $c = |S_{T_{i,\,m}}^{\ell}| = m - \ell + 1$.

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:

$$\boldsymbol{T} = [\{T^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{1.3}$$

$$\boldsymbol{T}_{i,\,m} = [\{T_{i,\,m}^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{1.4}$$

$$\boldsymbol{t}_i = [\{t_i^{(k)}\}_{k=1}^{d}]^{\intercal}. \tag{1.5}$$

Подпоследовательности временного ряда $\boldsymbol{T}$ можно разделить на два подмножества: множество *полных* подпоследовательностей, не содержащих пропущенных значений $\boldsymbol{S}_{\boldsymbol{T}}^{m}$ и множество *неполных* подпоследовательностей, содержащих хотя бы одно пропущенное значение:

$$\begin{aligned} \boldsymbol{S}_{\boldsymbol{T}}^{m} &= \left\{ \boldsymbol{T}_{i,m} \mid \forall t_j \in T_{i,m}^{(k)},\ t_j \neq \texttt{NaN} \right\}, \\ \overset{\circ}{\boldsymbol{S}}_{\boldsymbol{T}}^{m} &= \left\{ \overset{\circ}{\boldsymbol{T}}_{i,m} \mid \exists\ \overset{\circ}{t}_j \in \overset{\circ}{T}_{i,m}^{(k)},\ \overset{\circ}{t}_j = \texttt{NaN} \right\}, \end{aligned} \tag{1.6}$$

где $\overset{\circ}{\boldsymbol{T}}_{i,m}$ представляет собой подпоследовательность, в которой есть хотя бы одно пропущенное значение.

В дополнение к полным и неполным подпоследовательностям временного ряда $\boldsymbol{T}$ введем понятие восстановленной подпоследовательности $\overset{\bullet}{\boldsymbol{T}}_{i,m}$. Восстановленной называется такая подпоследовательность, которая была получена из неполной путем замены всех пропущенных значений на синтетические. Обозначим множество всех восстановленных подпоследовательностей как $\overset{\bullet}{\boldsymbol{S}}_{\boldsymbol{T}}^{m}$:

$$\overset{\bullet}{\boldsymbol{S}}_{\boldsymbol{T}}^{m} = \left\{ \overset{\bullet}{\boldsymbol{T}}_{i,m} \mid \forall \overset{\circ}{t}_j = \texttt{NaN}, \quad \overset{\bullet}{t}_j \neq \texttt{NaN}, \quad \overset{\circ}{t}_j \in \overset{\circ}{T}_{i,m}^{(k)}, \quad \overset{\bullet}{t}_j \in \overset{\bullet}{T}_{i,m}^{(k)} \right\}. \tag{1.7}$$

1.1.2. Поведенческие шаблоны временного ряда

Сниппеты представляют собой подпоследовательности временного ряда, выражающие типичные активности субъекта, деятельность которого описывает данный ряд, и определяются следующим образом [66].

Для заданной длины подпоследовательности m представим временной ряд T как набор не перекрывающихся подпоследовательностей, *сегментов*. Поскольку $m \ll n$, то без ограничения общности полагаем, что n кратно m, и набор сегментов Seg_T^m определяется следующим образом:

$$Seg_T^m = \{Seg_i\}_{i=1}^{n/m}, \quad Seg_i = T_{m\cdot(i-1)+1,\,m}. \tag{1.8}$$

Сниппеты T выбираются из Seg_T^m. Введем положительное целое число K ($1 \leqslant K \leqslant {}^n/_m$), которое представляет собой ожидаемое количество сниппетов (типичных активностей исследуемого субъекта). Определим, C_T^m, набор сниппетов длины m следующим образом:

$$C_T^m = \{C_i\}_{i=1}^K, \quad C_i \in Seg_T^m. \tag{1.9}$$

Сниппет имеет следующие атрибуты: индекс, набор ближайших соседей и покрытие. Для сниппета $C_i \in C_T^m$ указанные параметры обозначаются как $C_i.index$, $C_i.NN$ и $C_i.frac$ соответственно.

Индекс сниппета представляет собой номер сегмента, которому соответствует сниппет:

$$C_i.index = j \Leftrightarrow Seg_j = T_{m\cdot(j-1)+1,\,m}. \tag{1.10}$$

Метки классов определяются как индексы сниппетов в упорядоченном наборе C_T^m. Каждой подпоследовательности, входящей в множество ближайших соседей некоторого сниппета $C_i \in C_T^m$, сопоставляется класс i, соответствующий индексу этого сниппета в наборе.

Ближайшие соседи сниппета — это набор подпоследовательностей ряда, наименее удаленных от данного сниппета:

$$\begin{gathered} C_i.NN = \{T_{j,\,m} \mid Seg_{C_i.index} = \arg\min_{1 \leqslant s \leqslant {}^n/_m} \mathrm{MPdist}(T_{j,\,m}, Seg_s), \\ 1 \leqslant j \leqslant n - m + 1\}, \end{gathered} \tag{1.11}$$

где $\mathrm{MPdist}(\cdot,\cdot)$ — это функция расстояния, основанная на евклидовой нормированной метрике, предложенная в работе [52]. Покрытие сниппета — это отношение мощности множества его ближайших соседей к общему числу подпоследовательностей ряда соответствующей длины:

$$C_i.frac = \frac{|C_i.NN|}{n - m + 1}. \tag{1.12}$$

В наборе C_T^m сниппеты упорядочиваются в порядке убывания их покрытия:

$$\forall C_i, C_j \in C_T^m : \quad i < j \Leftrightarrow C_i.frac \geqslant C_j.frac. \tag{1.13}$$

Для многомерного временного ряда $\boldsymbol{T}$ назовем *словарем сниппетов* множество $\boldsymbol{C}_{\boldsymbol{T}}^m$, объединяющее в себе наборы сниппетов по всем измерениям ряда:

$$\boldsymbol{C}_{\boldsymbol{T}}^m = \bigcup_{k=1}^{d} C_{T^{(k)}}^m. \tag{1.14}$$

1.2. Жизненный цикл нейросетевых моделей

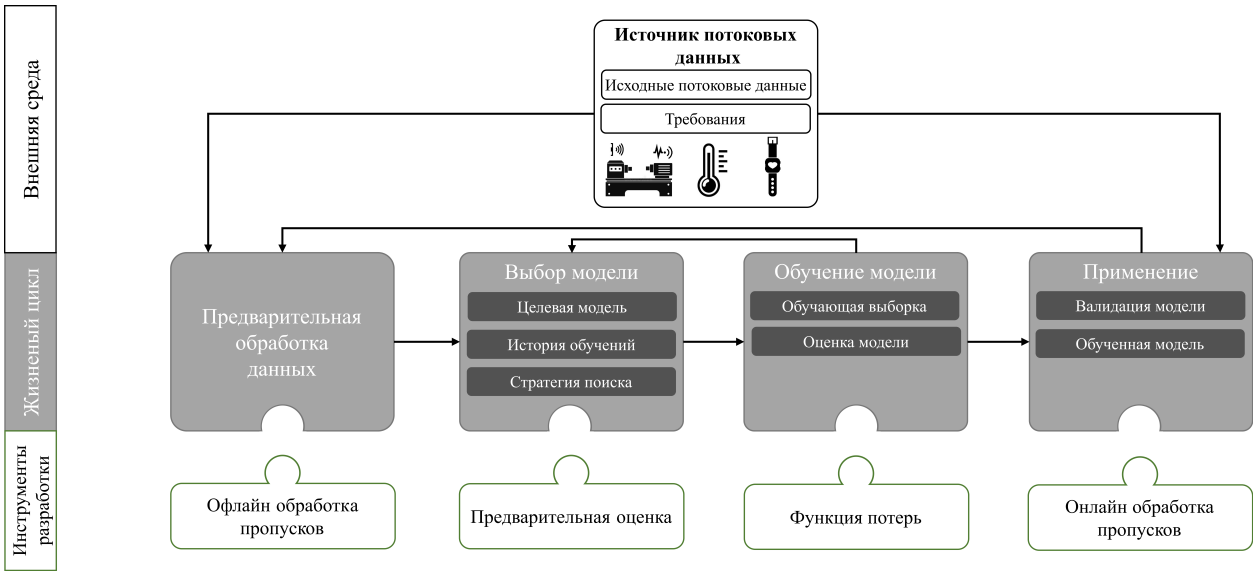


Рис. 1.1. Жизненный цикл моделей машинного обучения

Современные подходы к восстановлению потоковых данных, представленных в форме временных рядов, используются в интеллектуальных системах машинного обучения как инструментальные средства разработки. В рамках таких систем все процессы организуются как замкнутый жизненный цикл (см. рис. 1.1) [18], включающий последовательные этапы, каждый из которых выполняет определенную функцию и обеспечивает корректность дальнейшей обработки данных. Важно отметить, что при использовании нейросетевых методов восстановления на любом из этапов процесс

их разработки и внедрения целесообразно рассматривать как отдельный, вложенный жизненный цикл. Такой вложенный цикл воспроизводит общую структуру, соответствующую этапам, представленным на рис. 1.1, но применяемую к решению задачи восстановления пропусков. Жизненный цикл включает следующие ключевые этапы: определение исходных данных и системных требований, предварительную обработку данных, выбор нейросетевой модели, обучение и применение. Рассмотрим каждый этап более подробно.

Определение исходных данных и системных требований. Жизненный цикл нейросетевых моделей начинается со сбора исходных данных и формирования совокупности требований, определяющих особенности всех последующих этапов. На данном этапе выполняется анализ свойств данных, их полноты, качества и особенностей предметной области. Определяются задачи, решаемые в рамках жизненного цикла и формирующие основные требования. Полученные требования влияют на выбор ключевых компонентов последующих этапов. Например, если разрабатываемая нейросетевая модель предназначена для интеграции в систему реального времени, на этапе выбора архитектуры нейросетевой модели в стратегию поиска могут быть заложены ограничения, связанные с допустимой задержкой обработки. Если данные изменяют свои характеристики во времени или поступают из различных предметных областей, в качестве базовых могут рассматриваться нейросетевые модели, устойчивые к концептуальному дрейфу (concept drift) и способные поддерживать качество прогнозов при изменении статистических свойств входных данных [149].

Предварительная обработка данных. На этом этапе исходные данные подвергаются очистке, нормализации и другим преобразованиям, направленным на подготовку к их использованию для обучения модели. В процессе обработки устраняются шумы и аномалии, заполняются пропуски, данные приводятся к единому формату. При необходимости извлекаются новые признаки из исходных данных. Данные разделяются на обу-

чающие, валидационные и тестовые выборки, используемые на следующих этапах жизненного цикла.

В случае, если исходные данные содержат пропуски, на этапе предварительной обработки в качестве инструментальных средств разработки могут применяться *методы офлайн-восстановления временных рядов*. Точность восстановления влияет на весь жизненный цикл, поскольку некорректно восстановленные данные изменяют исходное распределение. В результате нейросетевая модель будет адаптироваться под восстановленные значения, снижая точность на полных данных. При этом выбор архитектуры нейросетевой модели может сместиться в сторону моделей, устойчивых к пропускам, точность прогнозов на исходных данных которых может быть ниже.

Выбор нейросетевой модели. На этом этапе определяется архитектура нейросетевой модели, которая будет использоваться для решения задач, сформулированных на этапе определения исходных данных и системных требований. Одними из ключевых элементов данного этапа являются целевая нейросетевая модель, история обучения и стратегия поиска. Целевая нейросетевая модель задает начальную архитектуру и служит отправной точкой для подбора конфигурации, обеспечивающей решение поставленной задачи с необходимой точностью. История обучения представляет собой накопленный опыт предыдущих экспериментов с различными архитектурами целевой модели.

Стратегия поиска определяет способ перебора архитектур и гиперпараметров для нахождения наиболее точной целевой модели. Полный перебор всех возможных конфигураций обычно невозможен или требует колоссальных вычислительных ресурсов, поскольку пространство поиска архитектур и гиперпараметров чрезвычайно велико [42, 43]. В связи с этим одними из основных инструментальных средств разработки на этом этапе являются *методы предварительной оценки*. Они позволяют оценить предварительную точность кандидатов на базе ограниченного набора данных, состоящего из результатов предыдущих итераций обучения. Качество предварительной оценки напрямую влияет на эффективность всего жизненного цикла:

занижение точности перспективных архитектур приводит к исключению потенциально качественных нейросетевых моделей, завышение точности слабых архитектур ведет к неоправданным затратам вычислительных ресурсов на их полное обучение.

В контексте задач восстановления временных рядов на этом этапе в качестве целевой нейросетевой модели выступают архитектуры, предназначенные для восстановления временных рядов в режимах онлайн и офлайн. В связи с этим на методы предварительной оценки накладываются дополнительные требования, связанные с необходимостью корректного сопоставления особенностей нейросетевых моделей с точностью восстановления. Такие методы в своих оценках должны учитывать влияние различных типов слоев, специализированных для обработки временных данных, на качество восстановления.

Обучение нейросетевой модели. На этом этапе осуществляется изменение параметров выбранной нейросетевой модели с целью минимизации ошибки предсказаний относительно целевых значений. Основными инструментальными средствами данного этапа являются *функция потерь и методы оценки*, которые позволяют измерять качество модели и управлять процессом обучения.

Функция потерь определяет, каким образом будут изменяться веса нейросетевой модели в процессе обучения. Если она задана некорректно или не соответствует задаче, решаемой в рамках жизненного цикла, процесс оптимизации теряет практическую ценность [125]. Нейросетевая модель может демонстрировать высокие показатели на обучающих данных, но оставаться неспособной эффективно решать целевую задачу. Методы оценки используются для анализа качества модели после завершения процесса обучения. Они используются для проверки модели на соответствие критериям, установленным на этапе определения исходных данных и требований, оценки достижения необходимого уровня точности и определения возможности ее допуска к этапу применения. В случае несоответствия модели требованиям потребуется возврат к предыдущим этапам жизненного цикла для повтор-

ного выбора архитектуры, настройки гиперпараметров или модификации функции потерь.

В контексте восстановления временных рядов как функция потерь, так и методы оценки должны учитывать специфику последовательной структуры данных. В отличие от задач, где качество определяется ошибкой на отдельных независимых наблюдениях, восстановление временного ряда требует оценки согласованности и точности всего восстановленного ряда. Функция потерь должна корректно отражать качество восстановления отдельных точек и степень поведенческой схожести восстановленного временного ряда с исходным, включающую такие аспекты, как сохранение временных зависимостей, динамики изменений и общих шаблонов поведения.

Применение модели. На данном этапе обученная нейросетевая модель интегрируется в целевую систему и используется для обработки поступающих данных в режиме онлайн. В процессе эксплуатации осуществляется непрерывный мониторинг, включающий контроль стабильности предсказаний и анализ изменения статистических свойств входных данных (концептуальный дрейф, Concept Drift [94]). На основе результатов мониторинга принимаются решения о необходимости повторного запуска предыдущих этапов жизненного цикла, таких как повторное обучение модели на актуальных данных или уточнение ее архитектуры.

Если специфика задачи предполагает наличие пропусков во входных данных, на этапе применения используются *методы онлайн-восстановления временных рядов*. В отличие от офлайн-методов, применяемых на этапе предварительной обработки, онлайн методы обеспечивают оперативное заполнение пропусков непосредственно в процессе работы системы, формируя корректные входные последовательности для целевой модели. Эффективность онлайн-восстановления критически важна для поддержания качества работы модели, так как ошибки на этом этапе непосредственно влияют на каждое принимаемое решение и могут иметь каскадный эффект. Незначительное искажение, внесенное на стадии онлайн-восстановления данных,

инициирует лавинообразное накопление последующих ошибок в различных компонентах системы. Например, некорректно восстановленное значение временного ряда приводит к ошибочному предсказанию модели. В результате происходит неверное управляющее воздействие или аналитическое заключение, изменяющее состояние целевой системы и приводящее к появлению новых аномалий. Таким образом формируется замкнутый цикл нарастающих искажений, который способен привести к существенному снижению устойчивости и деградации всей системы.

Анализ представленных этапов жизненного цикла позволяет выделить ключевые концептуальные и соответствующие им инструментальные средства, определяющие точность решения задачи восстановления временных рядов.

1. Методы офлайн и онлайн восстановления временных рядов, обеспечивающие формирование корректных входных последовательностей как на стадии предварительной обработки, так и в процессе применения нейросетевой модели.

2. Методы предварительной оценки нейросетевых моделей восстановления, используемые на этапе выбора архитектуры и позволяющие находить перспективные нейросетевые модели без их полного обучения.

3. Функции потерь, ориентированные на задачи восстановления временных рядов, формирующие критерий оптимизации и определяющие, какие свойства восстановленного ряда будут улучшаться в процессе обучения.

Таким образом, каждый из выделенных инструментов оказывает прямое влияние на точность и устойчивость нейросетевых моделей восстановления временных рядов. Поскольку каждый этап жизненного цикла формирует входные данные для следующего, инструментальные средства различных стадий образуют взаимосвязанную цепочку, в которой ошибки или неточности на любом этапе способны вызвать каскадный эффект и

привести к снижению качества итогового решения. Повышение точности инструментов на каждом этапе жизненного цикла является необходимым условием достижения высокого уровня качества восстановления.

В данной главе представлен обзор современных методов и нейросетевых моделей, используемых как инструментальные средства реализации различных этапов. В разделах 1.3 и 1.4.1 рассмотрены современные методы и архитектуры нейросетевых моделей, применяемые для онлайн и офлайн восстановления пропусков. Раздел 1.4.2 содержит описание функций потерь, используемых при обучении нейросетевых моделей восстановления временных рядов. Методы, применяемые для решения задачи предварительной оценки нейросетевых моделей, представлены в разделе 1.4.3.

1.3. Таксономия методов восстановления потоковых данных, представленных в форме временных рядов

Потоковые данные, представленные в форме временных рядов, можно рассматривать как хронологически упорядоченную последовательность наблюдений, отражающую поведение некоторого процесса во времени [30, 46, 101]. Наблюдение представляет собой элемент, содержащий значения исследуемых переменных, отражающие характеристики объекта, события или состояния системы в определенный момент времени. Например, в случае погодного временного ряда наблюдением можно считать вектор, включающий температуру, влажность и скорость ветра, зафиксированные в конкретный момент времени. Однако на практике часть значений может отсутствовать по различным причинам, включая ошибки измерений, сбои сенсоров или ограничения процесса сбора данных.

В работе [40] восстановление формализуется с помощью двух видов данных: полных и неполных. *Полные данные* (*complete data*) — это совокупность наблюдений, в которой для анализируемого объекта зафиксированы значения всех переменных для каждого наблюдения. *Неполные данные* (*incomplete data*) представляют собой совокупность наблюдений, в ко-

торых часть значений отсутствует. *Неполные данные* можно рассматривать как результат отображения *полных данных*, при котором часть значений наблюдений или целые наблюдения помечаются как пропущенные. В этом контексте восстановление пропусков можно рассматривать как обратное отображение, в результате которого пропускам сопоставляются значения из полных данных. Так как прямое обратное отображение данных в реальных условиях невозможно, восстановление пропусков выполняется с помощью приближенного обратного отображения, которое для заполнения недостающих элементов использует оценки, вычисленные на основании наблюдаемых значений.

Восстановление временного ряда представляет собой частный случай общей задачи восстановления пропусков. В случае восстановления временного ряда предполагается, что недостающие значения зависят не только от доступных наблюдений, но и от их порядка во временном ряду. Современные подходы к восстановлению пропусков во временных рядах используют информацию о порядке наблюдений и могут включать модели, учитывающие автокорреляцию, сезонные колебания, тренды и др. [127]. Существуют различные подходы к классификации методов восстановления временных рядов. Рисунок 1.2 иллюстрирует один из распространенных подходов к классификации методов, основанный на их делении на аналитические и нейросетевые [12, 72, 76]. Рассмотрим каждую из указанных групп более подробно.

Аналитические методы основываются на математических и статистических моделях, которые используют формальные предположения о структуре данных и характере пропусков. Ключевыми преимуществами таких методов являются интерпретируемость и относительно невысокие вычислительные затраты. Однако их эффективность существенно снижается при наличии нелинейных зависимостей.

Нейросетевые методы представляют собой группу методов, в рамках которой нейросетевые модели с большим числом параметров обучаются выявлению нелинейных и скрытых зависимостей в данных. Подобные ме-

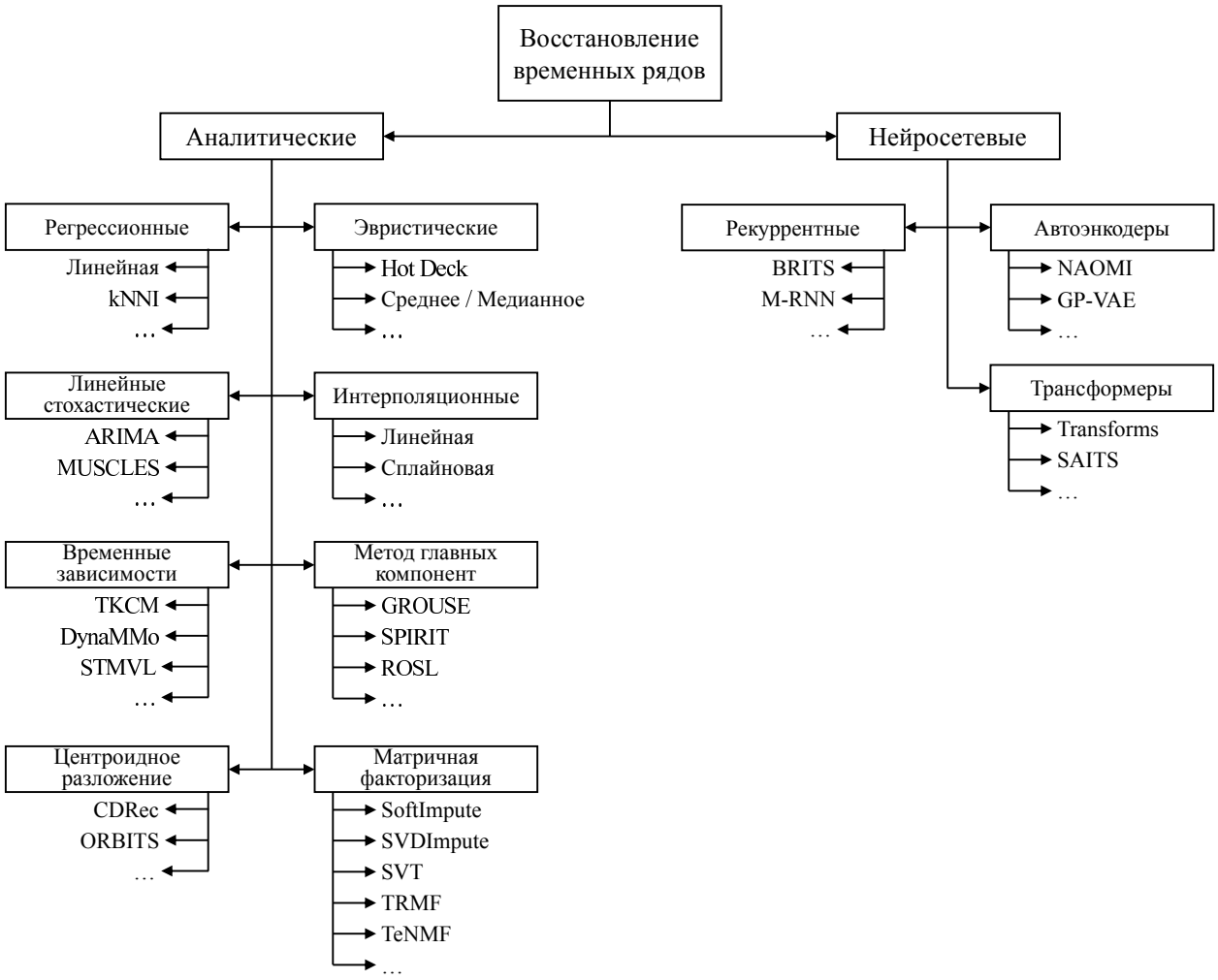


Рис. 1.2. Современные методы восстановления временных рядов

тоды характеризуются высокой адаптивностью и демонстрируют высокую точность при работе с большими и неоднородными многомерными наборами данных. К основным ограничениям нейросетевых подходов относятся значительные требования к вычислительным ресурсам, необходимость большого объема обучающих данных и низкая интерпретируемость получаемых результатов.

1.3.1. Аналитические методы

Аналитические методы могут быть классифицированы различными способами. Одним из таких способов является классификация по типу используемой модели и способу учета зависимости данных от времени [12, 47, 76].

Ниже представлены основные категории методов с описанием конкретных примеров.

Эвристические методы

Некоторые из базовых методов восстановления пропущенных значений основаны на элементарных эвристических правилах. В частности, метод Hot Deck предполагает замену пропущенного значения ближайшим по времени доступным наблюдением из того же временного ряда [28]. Альтернативный подход заключается в замене пропусков на различные характеристики ряда, такие как среднее арифметическое, медианное значение и др., вычисленным по всем доступным наблюдениям [22]. Несмотря на простоту реализации, указанные методы, как правило, характеризуются низкой точностью восстановления, особенно в условиях наличия сложных зависимостей в данных.

Интерполяционные методы

Интерполяционные методы восстанавливают пропущенные значения на основе информации о соседних наблюдениях временного ряда. Наиболее распространенными представителями данного класса являются линейная и сплайновая интерполяции [31]. В линейной интерполяции пропущенное значение вычисляется как линейная комбинация ближайших соседних точек, тогда как сплайновая интерполяция строит гладкую функцию, аппроксимирующую известные наблюдения. Подобные методы демонстрируют высокую точность при восстановлении отдельных или коротких серий пропусков. Однако при наличии продолжительных серий отсутствующих данных точность интерполяции существенно снижается. Кроме того, методы интерполяции не учитывают возможные зависимости между несколькими измерениями многомерного временного ряда.

Регрессионные методы

Регрессионные методы используют интерпретируемые модели, в которых задается зависимость недостающих значений от известных наблюдений. Во время восстановления модель может использовать как информацию, содержащуюся в самом временном ряду, так и внешние данные. Источниками внешних данных могут быть другие временные ряды, обладающие сильной корреляцией с восстанавливаемым рядом [150]. Коррелированные временные ряды широко применяются в метеорологических задачах, где пропущенные значения на одной станции могут быть оценены с использованием наблюдений, полученных на географически близких пунктах [106].

Одним из базовых регрессионных методов является линейная регрессия, которая предполагает существование линейной зависимости между доступными и пропущенными значениями. Более гибкий подход реализуется в методе восстановления на основе k ближайших соседей (k-Nearest Neighbors Imputation, kNNI) [22]. В рамках этого метода пропущенные значения оцениваются на основе наблюдений из k наиболее схожих подпоследовательностей временных рядов, определяемых с использованием выбранной метрики расстояния. В последующих модификациях данного метода [131] для повышения точности восстановления каждому соседу присваивается вес, пропорциональный евклидову расстоянию между восстанавливаемой подпоследовательностью и соответствующей соседней подпоследовательностью. Однако эффективность этого класса методов существенно снижается при наличии в данных сложной структуры или нелинейных зависимостей [150].

Линейные стохастические модели

Группа линейных стохастических моделей объединяет подходы, в которых текущее значение временного ряда описывается как линейная функция предыдущих наблюдений и случайной ошибки, характеризующей непред-

сказуемую часть процесса. В отличие от простых регрессионных моделей, данные методы учитывают временную зависимость и структуру ряда. Наиболее известным представителем данного класса является модель ARIMA (AutoRegressive Integrated Moving Average) [14, 15]. ARIMA модели позволяют эффективно восстанавливать пропуски в стационарных или приведенных к стационарности временных рядах. Однако практическое применение ограничивается необходимостью подбора гиперпараметров (p, d, q).

Для анализа потоковых данных были разработаны модификации линейных стохастических моделей, адаптированные к условиям последовательного поступления наблюдений. Одним из таких подходов является алгоритм MUSCLES (Multiple Streaming time series Completion using Least Squares Estimation) [150], основанный на авторегрессионной модели с рекурсивной оценкой параметров методом наименьших квадратов (Recursive Least Squares, RLS). Благодаря рекурсивной процедуре обновления параметров модель способна адаптироваться к изменениям структуры данных в реальном времени. Основным недостатком алгоритма MUSCLES является ограниченная масштабируемость. С увеличением числа измерений временного ряда существенно возрастают вычислительная сложность и объем используемой памяти.

Методы матричной факторизации

Методы матричной факторизации рассматривают задачу восстановления пропусков как задачу приближенного восстановления неполной матрицы, сформированной на основе временного ряда. Основная идея заключается в предположении о наличии в данных скрытой низкоранговой структуры, позволяющей представить исходную матрицу в компактной форме без существенной потери информации. Восстановление осуществляется путем разложения исходной матрицы в произведение матриц меньшей размерности с последующим итеративным уточнением приближенных значений. Такие методы, как правило, требуют начальной инициализации пропусков

и включают механизмы регуляризации для обеспечения устойчивости решения.

Базовым подходом из данной группы можно считать SVDImpute[131], который использует сингулярное разложение матрицы и восстанавливает пропущенные значения на основе первых k сингулярных компонент. Метод SoftImpute[97] развивает этот подход, применяя EM-алгоритм и мягкое пороговое обнуление сингулярных значений для автоматического выбора ранга. Алгоритм SVT (Singular Value Thresholding) [29] решает задачу минимизации ядерной нормы (nuclear norm) и восстанавливает матрицу путем итеративной пороговой фильтрации сингулярных значений.

Более сложные методы учитывают временную структуру данных. В частности, TRMF (Temporal Regularized Matrix Factorization)[154] расширяет стандартную факторизацию путем введения авторегрессионной регуляризации в скрытом пространстве признаков, представляя каждое скрытое значение как линейную комбинацию предыдущих состояний того же признака. Метод TeNMF[98] объединяет матричную факторизацию с агрегированием по временным окнам и регуляризацией.

Методы на основе центроидного разложения

В данную группу входят методы, основанные на центроидном разложении (Centroid Decomposition, CD) [74]. Центроидное разложение — это метод разложения матриц, который представляет исходную матрицу как произведение матрицы векторов «нагрузок» (loading) и матрицы значимостей (relevance). Векторы «нагрузок» вычисляются итеративно и подбираются таким образом, чтобы максимально отражать оставшуюся структурную информацию матрицы после учета предыдущих векторов.

Алгоритм CDRec [75] реализует классический итеративный подход к восстановлению пропусков на основе CD. Сначала пропущенные значения инициализируются, после чего выполняется аппроксимация пропусков с использованием ограниченного числа векторов нагрузки, отражающих основную структурную информацию матрицы. Далее оцененные значения

многократно уточняются через повторное разложение. Для задач потоковой обработки разработан метод ORBITS [73], который расширяет CDRec за счет инкрементального центроидного разложения (InCD), адаптированного к последовательному поступлению данных.

Методы на основе главных компонент

Методы данной группы используют проекцию данных в подпространства меньшей размерности. Ключевым элементом методов в данной группе является метод главных компонент (PCA) [118]. В контексте восстановления пропущенных значений PCA и его модификации применяются для проекции неполных наблюдений в низкоразмерное пространство с последующим восстановлением к исходной размерности. Особое внимание уделяется адаптациям метода к потоковым и онлайн-сценариям, где параметры модели обновляются в реальном времени. Обычно такие подходы используют градиентные или жадные алгоритмы оптимизации, обеспечивающие устойчивость к неполноте данных и изменчивости временного ряда.

Одним из примеров данной группы является метод GROUSE [21], который решает задачу восстановления без предварительной инициализации пропусков. Метод использует PCA для построения подпространства меньшей размерности. Известные значения исходной матрицы проецируются в это подпространство и обратно восстанавливаются в исходное измерение, образуя аппроксимацию исходных наблюдений. Стохастический градиентный спуск обновляет подпространство так, чтобы минимизировать расхождение между известными элементами данных и их аппроксимацией, после чего оцениваются пропущенные значения. Для обработки потоковых временных рядов разработан алгоритм SPIRIT [104]. Для каждой скрытой переменной строится авторегрессионная модель, параметры которой обновляются по мере поступления новых данных.

Методы на основе шаблонов

Данная группа включает методы, которые используют известные или выявляемые повторяющиеся шаблоны во временных рядах и зависимости между различными измерениями. Подходы данной группы опираются на предположение о наличии регулярной временной структуры и согласованности между источниками данных.

Алгоритм TKCM [140] предназначен для восстановления пропущенных значений путем выявления повторяющихся локальных шаблонов во множестве взаимосвязанных временных рядов. Метод предполагает, что в совокупности рядов присутствуют характерные структуры, регулярно повторяющиеся с определенной периодичностью.

Алгоритм DynaMMo [88] предназначен для восстановления пропущенных значений в группах коэволюционирующих временных рядов. В основе метода лежит совместное использование фильтров Калмана и EM-алгоритма. Фильтр Калмана оценивает состояние системы по данным с пропусками, используя информацию от связанных рядов. EM-алгоритм используется для итеративного уточнения пропущенных значений путем максимизации правдоподобия наблюдаемых данных.

Алгоритм STMVL [151] предназначен для восстановления пропущенных значений в пространственно-временных данных. Восстановленные данные строятся как агрегация выходов четырех моделей: двух глобальных статистических моделей, учитывающих пространственные и временные закономерности на основе исторических данных, и двух локальных алгоритмов коллаборативной фильтрации, обрабатывающих случаи нарушения общих закономерностей с опорой на недавние наблюдения.

1.3.2. Нейросетевые методы

Благодаря способности аппроксимировать сложные нелинейные функции [39, 63], нейросетевые модели являются эффективным инструментом для решения задач, связанных с временными рядами, включая прогноз [124],

классификацию [164], поиск аномалий [6] и др. В контексте восстановления пропущенных значений нейросетевые модели применяются для аппроксимации обратного отображения (см. раздел 1.3), которое по частично наблюдаемым данным формирует приближенные значения неполных наблюдений. Поскольку явная аналитическая формула для такого отображения отсутствует, его восстанавливают с помощью нейросетевых моделей, способных выявлять скрытые зависимости в данных. Под обучением в контексте нейросетевых моделей подразумевается процесс настройки параметров модели на основе имеющихся данных с целью минимизации ошибки между предсказанными и истинными значениями.

Методы данной группы обладают рядом существенных ограничений. Для эффективного обучения нейросетевых моделей требуется обширный и репрезентативный набор данных, включающий примеры и разнообразные варианты поведения временного ряда. Недостаток объема и разнообразия данных существенно снижает качество восстановления. Обучение нейросетей, особенно с использованием сложных архитектур, сопряжено с высокой вычислительной нагрузкой и длительным временем обучения, что ограничивает их практическое применение [121]. Несмотря на указанные недостатки, нейросетевые методы обладают рядом важных преимуществ. В частности, они способны выявлять скрытые структурные закономерности в данных, которые зачастую не могут быть смоделированы аналитическими методами. Гибкость архитектур и возможность их адаптации под специфические характеристики временных рядов позволяют обеспечивать высокую точность восстановления пропущенных значений в разнообразных задачах.

Существует множество подходов к классификации нейросетевых моделей, применяемых для восстановления пропущенных значений во временных рядах. Одним из наиболее распространенных является подход, основанный на архитектурных особенностях [12, 47, 129, 137]. Ниже рассматриваются основные классы моделей в рамках данной классификации.

Рекуррентные нейронные сети

Рекуррентные нейронные сети (recurrent neural network, RNN) представляют собой класс моделей, специально разработанных для обработки последовательных данных [115]. Ключевая особенность нейронных сетей данной архитектуры заключается в наличии скрытого состояния, которое обновляется на каждом шаге в зависимости от текущего значения и предыдущего состояния. В результате на каждом этапе обработки последовательности нейросетевая модель учитывает текущее значение и накапливает информацию, полученную на предыдущих шагах. Скрытое состояние выполняет роль «памяти», в которой накапливаются сведения о наблюдавшихся элементах ряда. Однако стандартные рекуррентные нейронные сети сталкиваются с трудностями при моделировании долгосрочных зависимостей, поскольку скрытое состояние на каждом временном шаге обновляется исключительно на основе состояния предыдущего шага и текущего значения. В результате нейросетевая модель проявляет повышенную чувствительность к информации, расположенной вблизи текущего временного шага, тогда как влияние данных, относящихся к более ранним моментам, постепенно ослабевает. При обучении на длинных последовательностях такое поведение приводит к возникновению проблемы исчезающего градиента. Нейронная сеть теряет способность адаптировать параметры на основе информации, содержащейся в удаленных временных шагах.

В отличие от стандартных RNN, GRU (Gated recurrent unit) использует механизм вентилей (gates) для управления потоком информации и определения ее важности на каждом шаге. Вентили позволяют модели учитывать контекст, эффективно обрабатывая как краткосрочные, так и долгосрочные зависимости. В сравнении с LSTM (Long Short-Term Memory) [62], который также решает проблему исчезающего градиента, GRU обладает более простой структурой и меньшим числом параметров (около 75 % от количества параметров LSTM при равных размерах входа и скрытого состояния). Благодаря этому модели на основе GRU отличаются высокой

вычислительной эффективностью по сравнению с LSTM и высокой точностью при работе с последовательными данными.

Существуют архитектурные модификации, в которых перед рекуррентными слоями применяются сверточные [69]. Сверточные слои выполняют предварительное извлечение локальных признаков. В результате снижается влияние шума и выбросов на качество восстановления. Входные данные преобразуются в обобщенные и структурированные представления, отражающие ключевые закономерности временного ряда. Рекуррентный слой модели получает входы, в которых сохранена ключевая динамика сигнала при сниженной чувствительности к шуму.

Автоэнкодеры

Автоэнкодеры представляют собой нейросетевые модели, обучающиеся извлекать компактные и информативные представления входных данных, сохраняя при этом их ключевую структуру в сжатой форме [20]. Автоэнкодер состоит из двух компонентов: Энкодера и Декодера. Энкодер преобразует входной сигнал в вектор скрытого состояния, обладающий меньшей размерностью по сравнению с исходными данными. Декодер восстанавливает данные из этого представления. В процессе обучения параметры модели настраиваются таким образом, чтобы формируемые векторы скрытого состояния содержали наиболее значимую информацию, необходимую для точного восстановления исходных данных на этапе декодирования.

В задачах восстановления временных рядов автоэнкодеры обучаются на последовательностях, содержащих пропуски, при этом пропущенные значения либо маскируются, либо заменяются специальными метками. Нейросетевая модель учится игнорировать эти неполные участки как нерелевантные для восстановления и формирует скрытое представление, опираясь на доступную информацию [163]. В процессе обучения автоэнкодер стремится приблизить скрытые представления неполных подпоследовательностей таким образом, чтобы они были максимально близки к представлениям аналогичных полных последовательностей.

Трансформеры

Трансформеры — это класс нейросетевых моделей, изначально разработанный для задач обработки последовательностей, таких как машинный перевод и анализ текста [132]. В отличие от рекуррентных нейронных сетей, трансформеры не используют последовательную обработку данных. Трансформеры используют механизм самовнимания (self-attention) [113], который позволяет модели учитывать взаимосвязи между элементами последовательности независимо от их положения. Ключевая идея заключается в том, что каждый элемент входной последовательности сопоставляется с остальными с помощью вычисления весов, определяющих степень влияния каждого элемента последовательности на текущую позицию. В задачах восстановления пропущенных значений трансформеры используют тот же механизм самовнимания для агрегации информации из наблюдаемых частей последовательности [141].

1.4. Обзор близких работ

1.4.1. Нейросетевые методы восстановления потоковых данных

Методы на основе рекуррентных нейронных сетей

Двунаправленная рекуррентная нейронная сеть BRITS (Bidirectional Recurrent Imputation for Time Series) [30] для восстановления многомерных временных рядов использует два слоя, состоящих 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.

В статье [92] предложена нейросетевая модель M-RNN, объединяющая два подхода к восстановлению данных: интерполяция и регрессия. Восстановление данных многомерного временного 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 измерения ряда и данные, полученные после интерполяции. Для случая одномерных временных рядов рассмотренная схема сводится к двунаправленной рекуррентной нейронной сети.

Методы на основе автоэнкодеров

В работе [92] описано восстановление многомерных временных рядов с помощью нейросетевой модели NAOMI (Non-AutOregressive Multiresolution Imputation). В NAOMI каждая нейросетевая модель, реализующая этапы кодирования и декодирования, представляет собой два слоя рекуррентных нейронов. Работа NAOMI по восстановлению пропущенных точек входной подпоследовательности выглядит следующим образом. Энкодер для каждой из двух крайних точек подпоследовательности формирует их скрытое представление. Декодер, используя полученные от Энкодера значения скрытого представления 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 центральной точки.

GP-VAE (Gaussian Process Variational AutoEncoder) [49] представляет собой нейросетевую модель, использующую для восстановления вариационный автоэнкодер (VAE) [77] и моделирование гауссовского процесса (Gaussian Process, GP) [111]. В отличие от автоэнкодера, VAE основан на вероятностной модели скрытых переменных. VAE кодирует входные дан-

ные в скрытое состояние, представленное в виде параметров многомерного распределения (математическое ожидание и дисперсия), определенных в скрытом пространстве. Авторы данной модели предполагают, что эволюцию скрытых переменных входных подпоследовательностей во времени можно представить как гауссовский процесс. Результатом моделирования гауссовского процесса является аппроксимация скрытого состояния входных данных. Аппроксимированное скрытое состояние поступает на вход декодера для формирования выхода модели.

Методы на основе трансформеров

Нейросетевая модель SAITS [44] предназначена для восстановления пропущенных значений в многомерных временных рядах. В основе модели лежит механизм самовнимания, реализованный через два диагонально-маскированных блока, которые позволяют эффективно захватывать временные зависимости и корреляции между признаками. Объединяя представления из этих блоков с учетом карты внимания и информации о пропущенных значениях, SAITS обеспечивает высокую точность восстановления данных без использования рекуррентных слоев.

Применение поведенческих шаблонов

В работах [66] предложено понятие *сниппетов* (*snippets*) — шаблонов временного ряда, отражающих типичные активности субъекта. Такие шаблоны позволяют выявлять повторяющиеся закономерности в данных и интерпретировать временной ряд как последовательность известных форм поведения. Использование сниппетов в контексте нейросетевых моделей восстановления может существенно повысить интерпретируемость и устойчивость алгоритмов, поскольку шаблоны могут служить дополнительными признаками, описывающими локальные особенности. Тем не менее, несмотря на потенциал данного подхода, современные нейросетевые модели восстановления не используют информацию о шаблонах временных рядов ни при формировании признакового пространства, ни на этапе обучения.

1.4.2. Функции потерь нейросетевых моделей восстановления потоковых данных

Для обучения нейросетевых моделей восстановления пропущенных значений во временных рядах используются, как правило, те же функции потерь, что и при решении задачи предсказания значения ряда на основе последовательности его предыдущих значений [68, 71, 135]. Наиболее часто применяются следующие функции потерь: средняя абсолютная ошибка (Mean Absolute Error, MAE) [109]; функции, основанные на среднеквадратичной ошибке (Mean Squared Error, MSE; Root Mean Squared Error, RMSE; Mean Squared Logarithmic Error, MSLE и др.); квантильная ошибка (Quantile Loss) [33]; функция потерь на основе логарифма гиперболического косинуса (Log-cosh Loss) [112]; функция потерь Хубера (Huber Loss) [123] и др.

Указанные функции ориентированы на оценку соответствия отдельных значений подпоследовательностей [71, 135] и не учитывают поведенческую схожесть. В последнее время исследователи предлагают функции потерь, которые позволяют учитывать форму и поведенческую схожесть подпоследовательностей временных рядов. Например, в работе [38] предложена функция потерь Soft-DTW, основанная на алгоритме динамической трансформации временной шкалы (DTW, Dynamic Time Warping) [25], которая позволяет учитывать временные искажения между подпоследовательностями. В работе [53] предложена функция MPdist для вычисления расстояния между подпоследовательностями временного ряда, учитывающая поведенческую схожесть.

Для сравнения указанных выше функций потерь рассмотрим, как меняются их значения в различных ситуациях изменения данных временного ряда (см. рис. 1.3). Пусть имеется временной ряд, представляющий собой значения синусоиды на отрезке $[0, 15]$, взятых с шагом 0.025, к точкам которого применяются три оператора, имитирующие различные ситуации, возникающие в процессе обучения нейросетевых моделей восстановления временных рядов: зашумление, масштабирование и сдвиг. Оператор зашум-

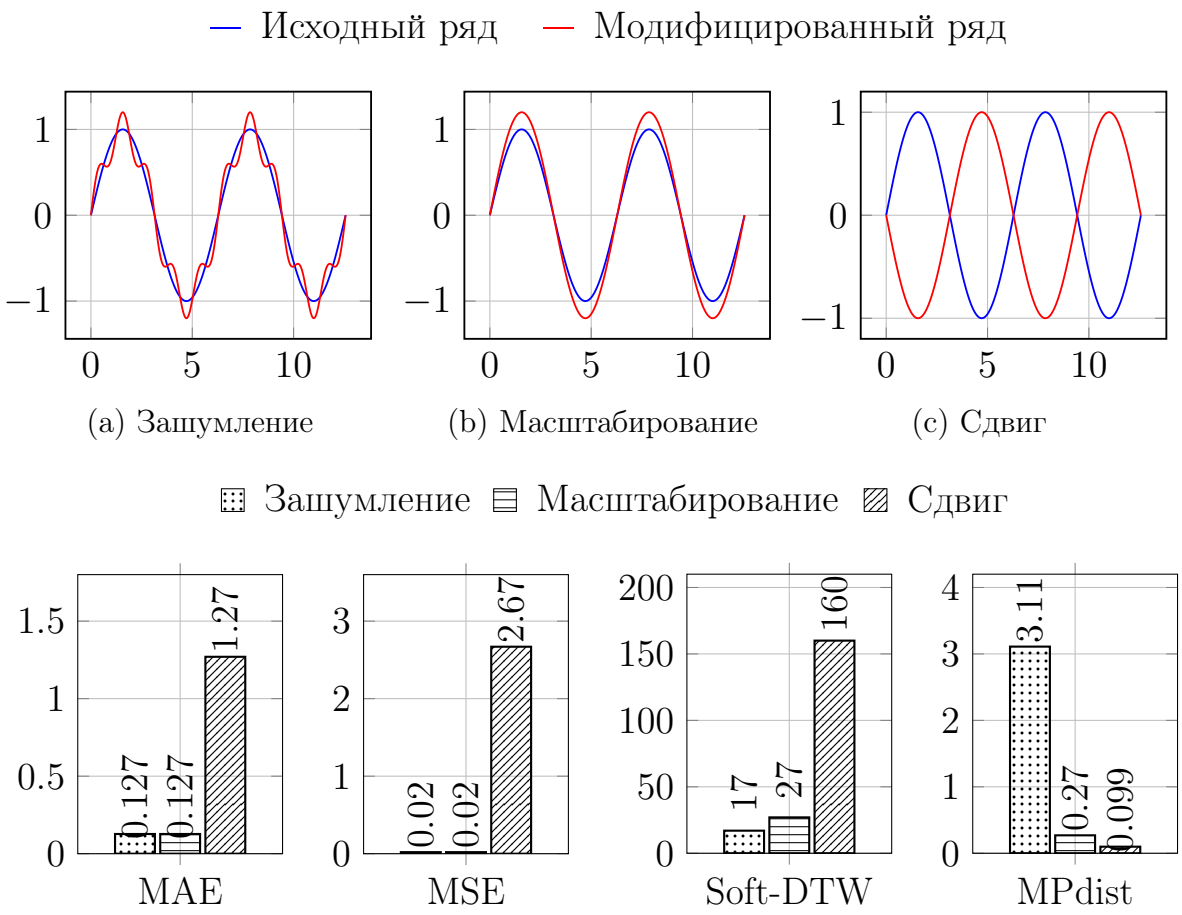


Рис. 1.3. Значения функций потерь, вычисленные после модификации временного ряда

ления 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. 1.3a, 1.3b и 1.3c соответственно. Применение оператора зашумления приводит к отсутствию всех рассмотренных выше видов схожести между исходным и результирующим рядами: поведенческое, по форме и по отдельным значениям. Оператор масштабирования приводит к потере схожести по отдельным значениям и форме (последняя частично), поведенческое сходство сохраняется. Наконец, оператор сдвига

сохраняет поведенческую схожесть, но нарушает сходство по отдельным значениям и форме.

Однако можно видеть (см. рис. 1.3d), 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 отдельным значениям.

1.4.3. Методы предварительной оценки нейросетевых моделей восстановления потоковых данных

В задачах поиска архитектуры нейронной сети (neural architecture search, NAS) для прогнозирования качества нейросетевых моделей применяются методы, которые можно разделить на три основные категории методов: ансамблевые, вероятностные и специализированные [51].

К ансамблевым методам можно отнести алгоритмы XGBoost, LightGBM и Random Forest [162]. Несмотря на их широкое применение, эффективность ансамблевых методов ограничена необходимостью трудоемкой настройки гиперпараметров и принципиальной неспособностью моделировать сложные иерархические зависимости, характерные для нейросетевых архитектур.

К вероятностным методам можно отнести следующие. Гауссовские процессы (gaussian processes, GP) применяются в NAS благодаря способности моделировать сложные нелинейные зависимости. В работе [128] представлен вариационный разреженный гауссовский процесс (Variational Sparse Gaussian Process, VSGP), адаптированный для входных данных с высокоразмерными пространствами признаков. Однако методы на основе GP остаются вычислительно затратными для крупномасштабных пространств поиска, содержащих десятки тысяч архитектур [152]. Байесовские методы, такие как байесовская линейная регрессия [143] и ее расширение DNGO [117], используют нейронные сети для преобразования признаков с последующей вероятностной оценкой прогнозируемых параметров.

Метод BOHAMIANN [119] применяет байесовские нейронные сети с использованием SGHMC (Stochastic Gradient Hamiltonian Monte Carlo) для оценки как ожидаемого значения, так и неопределенности прогноза. Многослойные перцептроны демонстрируют высокую обобщающую способность при наличии информативных признаковых описаний моделей [142]. Однако общим ограничением перечисленных подходов является отсутствие явного учета структурных зависимостей между слоями нейросетевой архитектуры, поскольку они опираются на векторные представления, формируемые методами типа one-hot кодирования.

К специализированным методам относятся следующие методы. Метод AutoCTS [147] предназначен для автоматизированного построения моделей, учитывающих пространственно-временные зависимости, через двухэтапный процесс поиска оптимальных модулей и их композиции. Аналогично, AutoTS [134] реализует двухэтапную стратегию сужения пространства поиска для эффективного проектирования моделей прогнозирования временных рядов. Тем не менее, существующие решения в области анализа нейросетевых моделей восстановления временных рядов не обеспечивают возможности оценки влияния структурных особенностей нейросетевых слоев на точность восстановления пропущенных значений.

Обзор показывает, что в настоящее время задача разработки нейросетевых моделей, методов и алгоритмов восстановления пропущенных данных

во временных рядах, является актуальной и остается предметом интенсивных научных исследований и практических разработок.

1.5. Выводы по главе 1

Рассмотрены основные этапы жизненного цикла нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Для каждого этапа выделены ключевые инструментальные средства их реализации, влияющие на точность восстановления. Проведен анализ влияния этих инструментов на качество восстановления, выявлены ограничения существующих подходов и обозначены направления для повышения эффективности решения задачи восстановления пропущенных значений. Проведена систематизация методологических подходов и выявлены ключевые тенденции в развитии данной предметной области. Введены основные формальные определения и обозначения, используемые в дальнейшем.

Аналитические методы, обладая высокой интерпретируемостью и относительно низкими вычислительными затратами, демонстрируют высокую точность восстановления при работе с данными простой структуры, однако их эффективность снижается при наличии сложных нелинейных зависимостей.

Описаны особенности нейросетевых моделей, применяемых для решения задач восстановления потоковых данных. Проведенный обзор показал, что в настоящее время разработано множество нейросетевых подходов к восстановлению пропущенных значений в потоковых данных, представленных в форме временных рядов. Нейросетевые модели обеспечивают высокую точность восстановления, однако отличаются повышенными требованиями к вычислительным ресурсам, слабой интерпретируемостью и отсутствием механизмов для применения шаблонов временных рядов. Существующие нейросетевые модели, как правило, обучаются напрямую на исходных временных последовательностях и не используют информацию о шаблонах, которые могли бы повысить точность восстановления.

Рассмотрены основные функции потерь и методы предварительной оценки, используемые в жизненном цикле восстановления потоковых данных, представленных в форме временных рядов. Используемые функции потерь (MAE, MSE, Huber Loss и др.) в основном ориентированы на минимизацию ошибок восстановления отдельных значений, но не учитывают поведенческую схожесть подпоследовательностей. Современные исследования (Soft-DTW, MPdist) направлены на преодоление этого ограничения, однако до настоящего времени не предложено универсальной функции потерь, комплексно отражающей все типы сходства между временными рядами.

Отдельное направление исследований связано с прогнозированием точности нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Для решения этой задачи применяются ансамблевые и вероятностные методы, включая XGBoost, GP, DNGO, BOHAMIANN, AutoCTS и AutoTS. Тем не менее, существующие решения в области анализа нейросетевых моделей восстановления временных рядов не обеспечивают возможность оценки влияния структурных особенностей нейросетевых слоев на точность восстановления пропущенных значений.

Обзор демонстрирует, что разработка методов, нейросетевых моделей и алгоритмов, обеспечивающих повышение точности на всех этапах жизненного цикла нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, сохраняет высокую актуальность и продолжает оставаться предметом активных научных исследований.

Глава 2. Нейросетевые методы и модели восстановления потоковых данных, представленных в форме временных рядов

В данной главе предложены новые нейросетевые методы восстановления потоковых данных, представленных в форме временных рядов, которые могут использоваться в качестве инструментальных средств разработки различных этапов жизненного цикла нейросетевых моделей. Предлагаемые методы восстановления достигают повышенной точности за счет совместного использования шаблонов активностей и нейросетевых моделей. В качестве инструментальных средств разработки в главе предлагаются следующие методы. В разделе 2.1 представлено описание нейросетевого метода SANNI, предназначенного для восстановления потоковых данных, представленных в форме временных рядов, в режиме онлайн. Раздел 2.2 содержит описание нейросетевого метода SAETI, предложенного для восстановления потоковых данных, представленных в форме временных рядов, в режиме офлайн.

2.1. Восстановление в режиме онлайн

В данном разделе предложен новый метод SANNI (Snippet and Neural Network based Imputation), обеспечивающий восстановление пропущенных значений потоковых данных, представленных в форме временных рядов, в режиме онлайн. На вход метода поступает репрезентативный фрагмент многомерного временного ряда, не содержащий пропусков. В ходе выполнения метода осуществляется обучение двух нейросетевых моделей, обеспечивающих восстановление пропущенных значений в подпоследовательностях. Для повышения точности восстановления в методе используются поведенческие шаблоны, содержащие основную информацию об активностях ряда. Приведено описание архитектур нейросетевых моделей, используемых в процессе восстановления. Нейросетевая модель Распознаватель выполня-

ет классификацию входной подпоследовательности. Нейросетевая модель Реконструктор осуществляет восстановление пропущенных значений, используя исходные данные и шаблонную подпоследовательность, сформированную на основе результатов классификации.

2.1.1. Нейросетевой метод SANNI

На вход метода SANNI поступает *репрезентативный фрагмент* многомерного временного ряда, содержащего d измерений. Под *репрезентативным фрагментом* понимается непрерывный участок длиной n, в котором отсутствуют пропущенные значения. Предполагается, что фрагмент содержит точки, которые были получены во время всех основных активностей субъекта. В дальнейшем такой фрагмент будет обозначаться как $\boldsymbol{T}$.

В качестве субъекта могут выступать такие сущности как человек, промышленное оборудование, техническая система и др. Например, если субъектом является человек, активностями будут считаться такие действия как ходьба, бег, прыжки и др. В этом случае измерениями будут считаться данные, полученные от носимых сенсоров: ускорение по трем осям, частота сердечных сокращений, количество шагов и другие параметры. Если субъектом выступает промышленный станок, то активностями могут быть запуск, процесс резки, смена инструмента и остановка. Измерения могут включать ток двигателя, температуру компонентов, частоту вращения и нагрузку на исполнительные узлы. В случае с технической системой, например, системой климат-контроля, активностями служат режимы работы: охлаждение, обогрев или ожидание. Измерениями в этом случае будут температура и влажность воздуха, ток компрессора, состояние вентиляторов и др.

Другим входным параметром метода является значимая длина подпоследовательности m. Предполагается, что эксперт в предметной области заранее определяет значимую длину таким образом, чтобы подпоследовательности заданной длины охватывали характерные и информативные части одной активности. Например, если данные получены с акселеромет-

ра, установленного на ноге человека во время различных активностей, то длина m может соответствовать одному или нескольким полным циклам движения ноги при ходьбе. В случае анализа работы промышленного станка подпоследовательность длиной m может включать полный цикл обработки: подвод инструмента, обработка материала и извлечение заготовки. Для системы климат-контроля длина m может соответствовать периоду включения и стабилизации температуры в режиме охлаждения. В задачах мониторинга сердечного ритма подпоследовательность длины m может включать несколько сердечных сокращений.

Репрезентативный фрагмент поступает на вход метода SANNI. В результате работы метода будет обучен набор нейросетевых моделей, последовательное применение которых позволит восстановить пропуски в подпоследовательностях временного ряда. Ниже в разделе 2.1.2 представлено описание компонентов метода, включая описание нейросетевых моделей, используемых для восстановления. Процесс обучения нейросетевых моделей, реализующих этапы обработки данных, описан в разделе 2.1.3. Раздел 2.1.4 посвящен способу применения обученных моделей для восстановления временных рядов.

2.1.2. Компоненты метода

На рис. 2.1 представлен метод SANNI. Предлагаемый метод включает две фазы: предобработка и восстановление подпоследовательностей. Фаза предобработки включает следующую последовательность этапов. На первом этапе *репрезентативный фрагмент* $\boldsymbol{T}$ поступает на вход Препроцессору, который выполняет предварительную обработку входных данных. В результате работы Препроцессора формируются следующие объекты: обучающие выборки для нейросетевых моделей метода и словарь шаблонов ряда $\boldsymbol{C}_T^m$ (см. формулу 1.14). На втором этапе, используя сформированные выборки, обучаются две нейросетевые модели: Распознаватель и Реконструктор. Нейросетевая модель Распознаватель обучается выполнять классификацию активностей в каждом измерении входной многомерной

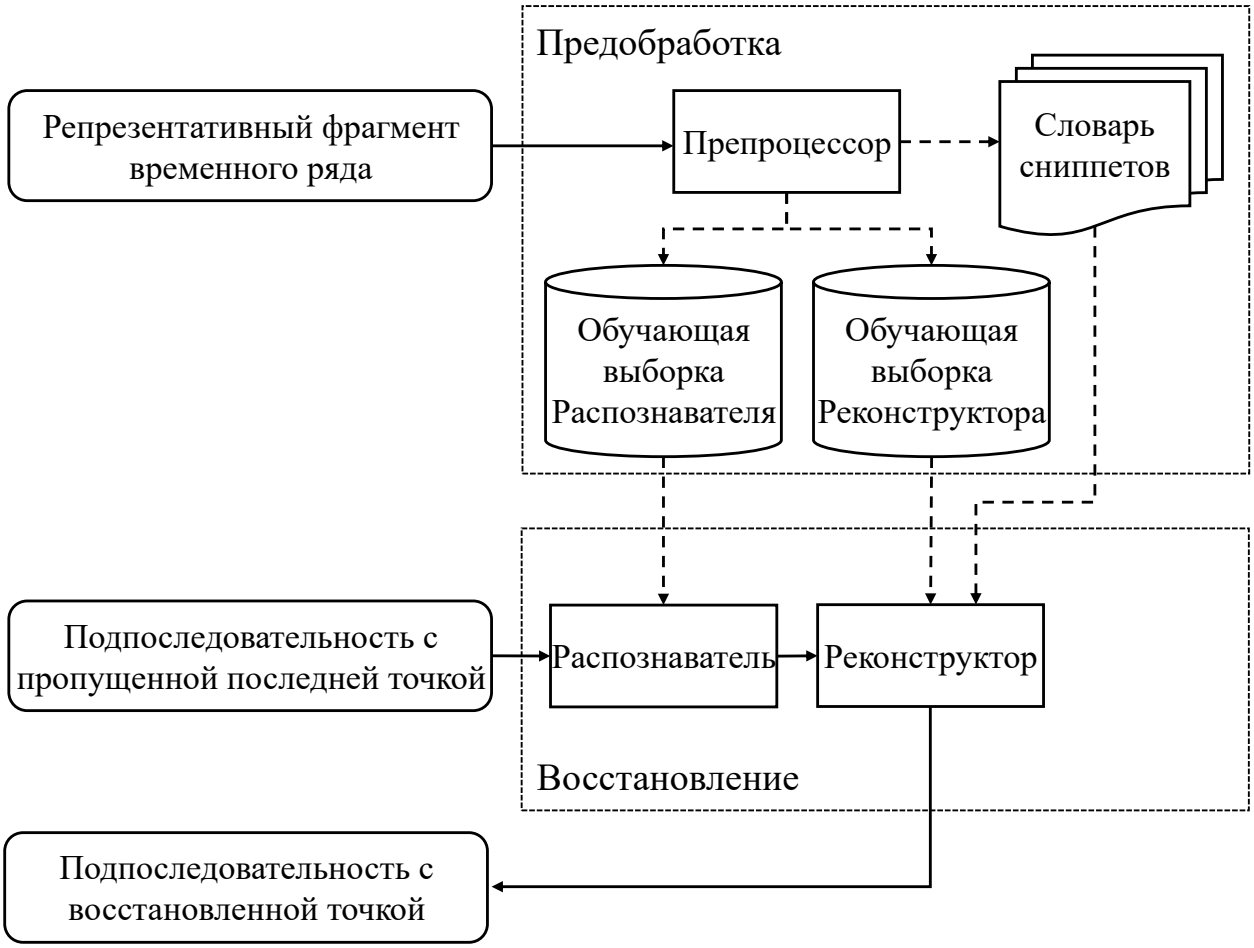


Рис. 2.1. Метод восстановления временного ряда в режиме онлайн

подпоследовательности. Нейросетевая модель Реконструктор, в свою очередь, обучается восстанавливать пропущенные значения, используя в качестве входных данных подпоследовательность с пропусками и шаблонную подпоследовательность. Под шаблонной подпоследовательностью в данном случае подразумевается такая подпоследовательность, в которой каждое измерение содержит шаблон, соответствующий распознанной активности.

Фаза восстановления пропущенных значений реализуется посредством последовательного применения обученных нейросетевых моделей к каждой подпоследовательности, содержащей пропуски.

Нейросетевая модель Распознаватель

Поскольку каждый шаблон C_i обобщает характерный тип поведения [66], наблюдаемый во время одной из активностей временного ряда, и ассоци-

ирован с группой схожих подпоследовательностей (см. формулу 1.11), его можно трактовать как прототип класса. В этом контексте возникает естественное разбиение временного ряда на классы, основанное на близости подпоследовательностей к соответствующему шаблону. Данное разбиение семантически отражает принадлежность каждой подпоследовательности к определенному типу активности.

Для формального описания множества классов введем набор меток Ψ, который определяется как индексы шаблонов в упорядоченном наборе C_T^m. Каждой подпоследовательности, входящей в множество ближайших соседей некоторого шаблона $C_\psi \in C_T^m$, может быть сопоставлена метка ψ, соответствующая индексу шаблона в наборе. Формально множество меток может быть представлено следующим образом:

$$\Psi = \{\psi\}_{\psi=1}^{K}, \tag{2.1}$$

где K соответствует числу активностей.

Введем функцию *классификации подпоследовательности* f_{class}, которая устанавливает соответствие между элементами множества подпоследовательностей S_T^m и множеством меток шаблонов Ψ. Для каждой подпоследовательности $T_{i,m} \in S_T^m$ функция f_{class} сопоставляет метку $\psi \in \Psi$, если $T_{i,m}$ принадлежит множеству ближайших соседей шаблона $C_\psi \in C_T^m$. Предполагается, что каждая подпоследовательность входит в множество ближайших соседей строго одного шаблона, что обеспечивает однозначность сопоставления метки $\psi \in \Psi$ через функцию f_{class}. Формально отображение может быть определено следующим образом:

$$f_{class} : S_T^m \to \Psi,\ f_{class}(T_{i,m}) = \psi,\ T_{i,m} \in C_\psi.NN,\ \psi \in \Psi. \tag{2.2}$$

Рассмотрим многомерную подпоследовательность как набор одномерных подпоследовательностей, каждая из которых соответствует отдельному измерению. Для каждого измерения с помощью функции *классификации подпоследовательности* может быть получена метка класса ψ. Объ-

единяя метки классов, полученные для всех измерений многомерной подпоследовательности, формируется вектор. Каждый элемент вектора представляет собой метку класса для соответствующего измерения подпоследовательности. Такой вектор в дальнейшем будем называть *вектором меток* $\boldsymbol{\psi}$. Множество таких векторов для всех подпоследовательностей временного ряда $\boldsymbol{T}$ обозначим как $\boldsymbol{\Psi}$. Формально *вектор меток* $\boldsymbol{\psi}$ может быть записан следующим образом:

$$\begin{gathered}\boldsymbol{\Psi} = \{\boldsymbol{\psi}_i\}_{i=1}^{n-m+1}, \\ \boldsymbol{\psi}_i = \{\psi_i^{(k)}\}_{k=1}^{d}, \\ \psi_i^{(k)} = f_{class}(T_{i,m}^{(k)}), \\ \boldsymbol{T}_{i,m} \in \boldsymbol{S}_{\boldsymbol{T}}^m, 1 \leqslant i \leqslant n-m+1, 1 \leqslant k \leqslant d.\end{gathered} \tag{2.3}$$

Многомерная классификация подпоследовательности $\boldsymbol{f}_{\text{class}}$ представляет собой многомерное обобщение функции f_{class}. Функция $\boldsymbol{f}_{\text{class}}$ устанавливает соответствие между элементами множества многомерных подпоследовательностей $\boldsymbol{S}_{\boldsymbol{T}}^m$ и элементами множества векторных меток $\boldsymbol{\Psi}$:

$$\boldsymbol{f}_{\text{class}} : \boldsymbol{S}_{\boldsymbol{T}}^m \to \boldsymbol{\Psi}, \quad \boldsymbol{f}_{\text{class}}(\boldsymbol{T}_{i,m}) = \boldsymbol{\psi}_i, \quad \forall \boldsymbol{T}_{i,m} \in \boldsymbol{S}_{\boldsymbol{T}}^m. \tag{2.4}$$

Однако функция $\boldsymbol{f}_{\text{class}}$ определена только на множестве полных подпоследовательностей и не может быть напрямую применена к последовательностям, содержащим пропуски. Для преодоления этого ограничения в методе используется нейросетевая модель Распознаватель, которая моделирует поведение функции $\boldsymbol{f}_{\text{class}}$ на расширенном множестве подпоследовательностей $\overset{\bullet}{\boldsymbol{S}}_{\boldsymbol{T}}^m$, включающем полные и неполные подпоследовательности. В результате ее применения для каждой многомерной подпоследовательности, независимо от наличия в ней пропусков, формируется вектор прогнозных меток $\overset{\bullet}{\boldsymbol{\psi}} \in \overset{\bullet}{\boldsymbol{\Psi}}$. Функция $\overset{\bullet}{\boldsymbol{f}}_{\text{class}}$, которую реализует нейросетевая модель Распознаватель, устанавливает соответствие между элементами расширенного множества подпоследовательностей $\overset{\bullet}{\boldsymbol{S}}_{\boldsymbol{T}}^m$ и множеством векторов прогноз-

ных меток $\overset{\bullet}{\boldsymbol{\Psi}}$:

$$\overset{\bullet}{\boldsymbol{f}}_{\text{class}} : \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}} \to \overset{\bullet}{\boldsymbol{\Psi}}, \quad \overset{\bullet}{\boldsymbol{f}}_{\text{class}}(\boldsymbol{T}_{i,m}) = \overset{\bullet}{\boldsymbol{\psi}}_{i}, \quad \forall \boldsymbol{T}_{i,m} \in \boldsymbol{S}^{m}_{\boldsymbol{T}}. \tag{2.5}$$

Предполагается, что функция $\overset{\bullet}{\boldsymbol{f}}_{\text{class}}$, реализуемая нейросетевой моделью Распознаватель, совпадает с функцией $\boldsymbol{f}_{\text{class}}$ на множестве $\boldsymbol{S}^{m}_{\boldsymbol{T}}$:

$$\overset{\bullet}{\boldsymbol{f}}_{\text{class}}(\boldsymbol{T}_{i,m}) = \boldsymbol{f}_{\text{class}}(\boldsymbol{T}_{i,m}), \quad \forall \boldsymbol{T}_{i,m} \in \boldsymbol{S}^{m}_{\boldsymbol{T}}. \tag{2.6}$$

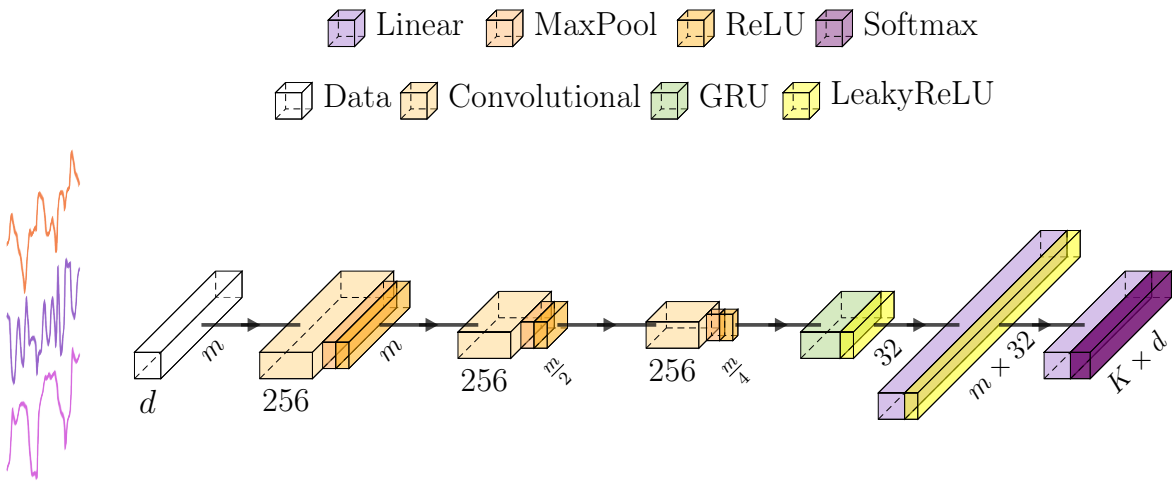


Рис. 2.2. Архитектура нейросетевой модели Распознаватель

На рис. 2.2 представлена архитектура нейросетевой модели Распознаватель. Согласно формуле 2.5, на вход модели поступает многомерная подпоследовательность $\boldsymbol{T}_{i,m}$, содержащая пропущенные значения. Предполагается, что пропуски расположены во всех значениях последней многомерной точки. Перед обработкой входных данных пропущенные значения заменяются на $\texttt{NIL} = -1$. Данная замена обусловлена необходимостью корректной обработки неполных данных в модели. Использование специального обозначения, такого как `NIL`, позволяет модели различать наблюдаемые значения и пропущенные элементы.

Входная подпоследовательность последовательно обрабатывается тремя сверточными, одним рекуррентным и двумя полносвязными слоями. В

результате обработки на выходе модели формируется матрица вероятностей $\boldsymbol{P} \in \mathbb{R}^{d \times K}$. Каждая i-я строка матрицы $\boldsymbol{P}$ представляет собой вектор вероятностей, где j-й элемент соответствует вероятности того, что i-е измерение входной подпоследовательности относится к j-му классу активностей. На основе матрицы вероятностей $\boldsymbol{P}$ формируется *вектор меток* $\overset{\bullet}{\boldsymbol{\psi}}$, который используется при подготовке шаблонной подпоследовательности.

Каждый сверточный слой содержит 256 фильтров с размером ядра 5, которые применяются к входной подпоследовательности для извлечения локальных признаков. После каждого сверточного слоя применяется операция подвыборки по максимальному значению (Max-pooling) с размером окна 2, которая уменьшает размер карт признаков вдвое. В качестве функции активации всех сверточных слоев используется функция ReLU [61].

В качестве рекуррентного слоя используется GRU [36] с размерностью вектора скрытого состояния, равной 32. Данный слой отвечает за анализ временных зависимостей между элементами входной последовательности. Рекуррентный слой извлекает признаки каждого временного шага, учитывая информацию с предыдущих временных шагов. В этом и последующих слоях в качестве функции активации используется Leaky ReLU [56], позволяющая избежать проблемы «умирающих нейронов» [95].

Следующие два полносвязных слоя формируют выход *Распознавателя*. Первый слой содержит $32 \cdot m$ нейронов и преобразует скрытое состояние GRU в векторное представление. Второй слой, состоящий из $d \cdot K$ нейронов, формирует выходную матрицу вероятностей $\boldsymbol{P} \in \mathbb{R}^{d \times K}$. В качестве функции активации второго слоя используется softmax [26], которая применяется к каждой строке выходной матрицы. В результате формируется матрица $\boldsymbol{P}$, где элемент $\boldsymbol{P}_i^j$ интерпретируется как вероятность принадлежности i-го измерения входной подпоследовательности к j-му классу активности. Вектор прогнозных меток $\overset{\bullet}{\boldsymbol{\psi}}$ формируется путем применения операции $\arg\max$ по столбцам к каждой строке матрицы вероятностей $\boldsymbol{P}$, выбирающей наиболее вероятный класс для соответствующего измерения:

$$\overset{\bullet}{\boldsymbol{\psi}}_i = \arg \max_{1 \leqslant j \leqslant K} \boldsymbol{P}_i^j, \quad \forall 1 \leqslant i \in \leqslant d. \tag{2.7}$$

На основе полученного нейройсетевой моделью вектора прогнозных меток $\overset{\bullet}{\boldsymbol{\psi}}$ формируется шаблонная подпоследовательность $\boldsymbol{F} \in \mathbb{R}^{m,d}$. Каждое измерение шаблонной подпоследовательности соответствует шаблону извлеченному из словаря $\boldsymbol{C}_T^m$. Шаблон выбирается на основе предсказанной метки класса для соответствующего измерения входной подпоследовательности:

$$\boldsymbol{F}^{(k)} = C_{\overset{\bullet}{\boldsymbol{\psi}}_k}, \; C_{\overset{\bullet}{\boldsymbol{\psi}}_k} \in C_{T^{(k)}}^m, \; C_{T^{(k)}}^m \in \boldsymbol{C}_T^m, \; 1 \leqslant k \leqslant d. \tag{2.8}$$

Нейросетевая модель Реконструктор

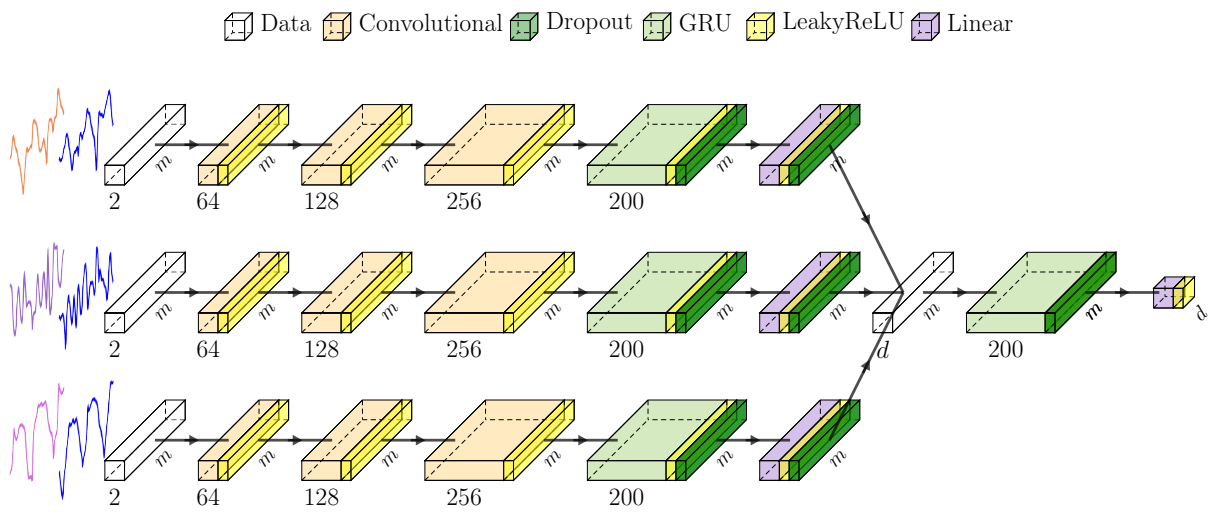


Рис. 2.3. Архитектура нейросетевой модели Реконструктор

На рис. 2.3 представлена архитектура нейросетевой модели Реконструктор. На вход модели поступает многомерная подпоследовательность временного ряда $\boldsymbol{T}_{i,m}$, у которой пропуски в последней многомерной точке были заменены на `NIL`, и шаблонная подпоследовательность $\boldsymbol{F}$. В результате обработки входных данных нейросетевой моделью формируется многомерная восстановленная точка $\overset{\bullet}{\boldsymbol{t}}$, которая представляет собой синтетическое значение пропущенной точки временного ряда. Во время обработки входных данных формируется набор $\boldsymbol{M}$, состоящий из d подпоследовательностей-макетов $\boldsymbol{M}_k \in \mathbb{R}^{m \times 2}$. Первое и второе измерение k-й подпоследовательности-

макета содержат k-е измерение $\boldsymbol{T}_{i,m}$ и $\boldsymbol{F}$ соответственно:

$$\boldsymbol{M}_k^1 = \boldsymbol{T}_{i,m}^k, \boldsymbol{M}_k^2 = \boldsymbol{F}^k. \tag{2.9}$$

На первом шаге обработки данных сформированный набор $\boldsymbol{M}$ обрабатывается набором из k независимых последовательностей слоев. Каждая последовательность преобразует одну подпоследовательность-макет в векторное представление. Независимая обработка каждого измерения временного ряда обусловлена тем, что в задачах многомерного восстановления каждое измерение имеет собственную семантику, отражающую уникальные характеристики отслеживаемого процесса. Например, в медицине одно измерение может отражать частоту сердечных сокращений, другое содержит информацию об уровне кислорода в крови и др. Несмотря на то, что между этими измерениями могут существовать связи, их шаблоны могут существенно различаться. Объединение разных измерений на раннем этапе может негативно отразиться на качестве восстановления, поскольку характерные шаблоны одного канала могут перекрывать ключевые признаки другого. Чтобы этого избежать, каждое измерение обрабатывается собственной последовательностью слоев, которые извлекает признаки, специфичные именно для этого измерения, в контексте его шаблона.

Каждая последовательность слоев включает три сверточных слоя, рекуррентный слой и полносвязный слой. Сверточные слои содержат 64, 128 и 256 фильтров с размером ядра, равным 5. В процессе обработки сверточные слои извлекают пространственные признаки, позволяющие модели учитывать структурные закономерности шаблона и подпоследовательности. Полученные признаки проходят через GRU слой с размером скрытого состояния, равным 200, который извлекает временные зависимости. Завершающим слоем является полносвязный слой с m нейронами, который формирует векторное представление подпоследовательности-макета.

После того как из каждой подпоследовательности-макета было извлечено ее векторное представление, выходы всех последовательностей объединяются в матрицу векторных представлений размером $m \times d$. Матрица

векторных представлений поступает на вход рекуррентному слою на основе GRU с размером скрытого состояния, равным 200. Рекуррентный слой обрабатывает объединенные векторные представления, позволяя модели извлекать признаки, отражающие зависимости между различными измерениями временного ряда. Выход из рекуррентного слоя передается на вход полносвязному слою, состоящему из d нейронов, который формирует окончательный результат модели — восстановленную многомерную точку $\overset{\bullet}{\boldsymbol{t}}$.

2.1.3. Обучение нейросетевых моделей

Для применения описанных выше нейросетевых моделей для восстановления многомерных подпоследовательностей требуется провести их обучение. Обучение нейронных сетей заключается в подборе параметров модели (весов и смещений), которые минимизируют ошибку между предсказанными и фактическими значениями целевой переменной. Обучение представляет собой итеративный процесс, состоящий из множества эпох. На каждой эпохе вычисляется значение функции потерь, измеряющей расхождение между предсказанными и реальными значениями. Во время обучения для каждой модели вычисляется собственная функция потерь. Функция потерь модели Распознаватель оценивает степень соответствия между векторами прогнозируемых меток $\overset{\circ}{\boldsymbol{\psi}}$ и векторами меток, полученных для множества $\overset{\circ}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$. Функция потерь модели Реконструктор оценивает расхождение между восстановленными многомерными точками $\overset{\bullet}{\boldsymbol{t}}$ и соответствующими истинными многомерными точками $\boldsymbol{t}$ последовательностей из множества $\boldsymbol{S}^{m}_{\boldsymbol{T}}$.

В процессе обучения модели используется обучающая выборка, состоящая из пар входных данных и соответствующих им целевых выходных значений. Для обеих моделей данного метода обучающая выборка формируется на основе фрагмента $\boldsymbol{T}$, который подвергается предварительной обработке.

Предварительная обработка

В процессе предварительной обработки каждое измерение временного ряда $\boldsymbol{T}$ подвергается нормализации. В результате нормализации каждое измерение $T^{(k)}$ временного ряда $\boldsymbol{T}$ масштабируется до диапазона $[0, 1]$ независимо. В качестве нормализации используется минимаксная нормализация:

$$\hat{t}_i^{(k)} = \frac{t_i^{(k)} - \min\limits_{1 \leqslant j \leqslant n} t_j^{(k)}}{\max\limits_{1 \leqslant j \leqslant n} t_j^{(k)} - \min\limits_{1 \leqslant j \leqslant n} t_j^{(k)}}, \quad 1 \leqslant k \leqslant d, \quad 1 \leqslant i \leqslant n. \tag{2.10}$$

На следующем шаге предварительной обработки для фрагмента $\boldsymbol{T}$ формируется словарь шаблонов $\boldsymbol{C}_T^m$. К каждому измерению многомерного фрагмента $\boldsymbol{T}$ применяется алгоритм поиска сниппетов PSF [168], в результате работы которого извлекается набор из K шаблонов длины m. Далее, к каждой многомерной подпоследовательности из фрагмента $\boldsymbol{T}$, применяется функция многомерной классификации $\boldsymbol{f}_{\text{class}}$ (см. формулу 2.4). В результате каждому элементу выборки сопоставляется соответствующий вектор меток $\boldsymbol{\psi} \in \boldsymbol{\Psi}$.

Формирование обучающих выборок

Каждая из нейросетевых моделей, используемых в предлагаемом методе, решает специфическую задачу и прогнозирует различные целевые переменные. В связи с этим для каждой из них формируется отдельная обучающая выборка, соответствующая ее функциональному назначению. В дальнейшем обучающую выборку будем обозначать как $D = \langle \boldsymbol{X}, \boldsymbol{Y} \rangle$, где $\boldsymbol{X}$ и $\boldsymbol{Y}$ представляют собой входные и выходные данные модели соответственно.

Входными данными модели Распознаватель полагаются многомерные подпоследовательности длиной m входного фрагмента $\boldsymbol{T}$, у которых последнее значение в каждом измерении было заменено на $\texttt{NIL} = -1$. Выходными значениями полагаются вектора меток $\boldsymbol{\psi}$, полученные для подпоследовательностей на этапе предварительной обработки. Формально обучающая выборка модели Распознаватель может быть представлена следующим

образом:

$$D_{\text{Recognizer}} = \{\langle \boldsymbol{X},\, \boldsymbol{Y}\rangle \mid X^{(k)} = T^{(k)}_{i,\,m-1} \cdot \texttt{NIL},\ \texttt{NIL} = -1,\ \boldsymbol{Y} = \boldsymbol{\psi}_i,\ \boldsymbol{\psi}_i = \boldsymbol{f}_{\text{class}}(\boldsymbol{T}_{i,m}),\ 1 \leqslant i \leqslant n-m+1,\ 1 \leqslant k \leqslant d,\ 1 \leqslant s \leqslant K\}, \tag{2.11}$$

где символ «$\cdot$» обозначает операцию конкатенации.

Входные данные модели Реконструктор представляют собой многомерные подпоследовательности длины m входного фрагмента $\boldsymbol{T}$, у которых последнее значение в каждом измерении было заменено на NIL. Выходными данными полагаются последние значения входных подпоследовательностей до замены на NIL:

$$D_{\text{Reconstructor}} = \{\langle \boldsymbol{X},\, \boldsymbol{Y}\rangle \mid X^{(k)} = T^{(k)}_{i,\,m-1} \cdot \texttt{NIL},\ \texttt{NIL} = -1,\ Y^{(k)} = t^{(k)}_{i+m},\ 1 \leqslant i \leqslant n-m+1,\ 1 \leqslant k \leqslant d\}, \tag{2.12}$$

где символ «$\cdot$» обозначает операцию конкатенации.

Вычисление ошибки

В качестве функции потерь для Нейросетевой модели Распознаватель используется кросс-энтропия (CrossEntropyLoss) [48], которая оценивает степень соответствия векторов предсказанных меток $\overset{\bullet}{\boldsymbol{\psi}}$ и векторов меток $\boldsymbol{\psi}$, полученных для многомерных подпоследовательностей временного ряда $\boldsymbol{T}$. Согласно определению выхода модели (см. формулу 2.7), вектор предсказанных меток $\overset{\bullet}{\boldsymbol{\psi}}$ формируется на основе матрицы вероятностей $\boldsymbol{P}$. Следовательно, вектор предсказанных меток будет совпадать с вектором меток $\boldsymbol{\psi}$ тогда, когда значения вероятностей, расположенные в позициях, соответствующих порядковым номерам истинных классов, являются максимальными в каждой строке матрицы $\boldsymbol{P}$. Для максимизации вероятностей в процессе обучения функция потерь вычисляется как среднее по строкам отрицательных логарифмов значений, расположенных в позициях истинных классов. Формально ошибка Распознавателя может быть представлена

следующим образом:

$$L_{\text{Recognize}}(\boldsymbol{P},\, \boldsymbol{Y}) = -\frac{1}{d}\sum_{k=1}^{d}\sum_{i=1}^{K} 1_{\{Y^{(k)}\}}(i)\cdot\log(\boldsymbol{P}_k^i),$$
$$1_{\{Y^{(k)}\}}(i) = \begin{cases} 1, & \text{если } i = Y^{(k)}, \\ 0, & \text{если } i \neq Y^{(k)}, \end{cases} \tag{2.13}$$

где $1_{\{Y^{(k)}\}}(i)$ представляет собой индикаторную функцию, которая принимает значение 1, если вероятность находится на позиции истинного класса, и 0 в противном случае.

В качестве функции потерь восстановления используется среднеквадратичная ошибка (MSELoss) [135], которая рассчитывается как среднее значение квадратов отклонений между предсказанными и истинными значениями. В рассматриваемом случае предсказанием модели является восстановленная многомерная точка $\dot{\boldsymbol{t}}$, тогда как в качестве эталонного значения используется последняя точка входной подпоследовательности. Соответственно, ошибка Реконструктора формально может быть представлена следующим образом:

$$L_{\text{Reconstructor}}(\dot{\boldsymbol{t}},\, \boldsymbol{Y}) = \frac{1}{d}\sum_{i=1}^{d}(Y^{(i)} - \dot{t}_i)^2. \tag{2.14}$$

2.1.4. Применение метода

Работа метода SANNI может быть описана следующим образом. Пусть обучающие выборки для нейросетевых моделей Распознавателя и Реконструктора сформированы на основе временного ряда $\boldsymbol{T}$, и соответствующие нейросетевые модели обучены на этих данных в соответствии с описанным выше в разделе 2.1.3.

Пусть задан временной ряд $\boldsymbol{U}$, который отражает активность того же субъекта, что и ряд $\boldsymbol{T}$, однако содержит пропущенные значения. В ходе восстановления выполняется просмотр ряда $\boldsymbol{U}$ с помощью скользящего окна длины m, начиная с первого элемента ряда. Если последняя точка $\boldsymbol{U}_{i,\,m}$ яв-

ляется пропущенной, то каждое измерение подпоследовательности $\boldsymbol{U}_{i,\,m-1}$ подвергается минимаксной нормализации для приведения ее значений к промежутку, границами которого являются минимум и максимум соответствующих измерений временного ряда $\boldsymbol{T}$. Нормализованная подпоследовательность подается на вход Распознавателя, после чего его выход в виде шаблонной подпоследовательности совместно с нормализованной подпоследовательностью $\boldsymbol{U}_{i,\,m-1}$ подаются 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 имеют место «холодного старта» (пропущена точка $\boldsymbol{u}_k \in \boldsymbol{U}_{1,\,m}$), то на вход нейросетевых моделей подается подпоследовательность, которая представляет собой конкатенацию «фантомной» и реальной частей ряда $\boldsymbol{U}$. «Фантомная» часть представляет собой многомерную подпоследовательность длины $m - k$, в которой каждое измерение заполнено медианными значениями, вычисленными по соответствующему измерению всего временного ряда $\boldsymbol{T}$. В качестве реальной части фигурируют точки $\{\boldsymbol{u}_i\}_{i=1}^{k-1}$ первой подпоследовательности ряда.

2.2. Восстановление в режиме офлайн

В данном разделе представлен новый нейросетевой метод SAETI (Snippet based Autoencoder for Timeseries Imputation) для восстановления пропусков в потоковых данных, представленных в форме временных рядов, в режиме офлайн. Предложенный метод имеет общий теоретический базис с методом SANNI. Общий базис заключается в совместном использовании шаблонов и нейросетевых моделей для восстановления временных рядов. Метод использует адаптированные к работе с пропусками нейросетевые архитектуры моделей Распознавателя и Реконструктора. Архитектура Реконструктора SAETI основана на автоэнкодере с дополнительным входом для поведенческих шаблонов.

2.2.1. Нейросетевой метод SAETI

В данном разделе представлено описание метода SAETI, включая используемые нейросетевые модели для восстановления данных. На вход метода поступает репрезентативный фрагмент многомерного временного ряда длиной n, содержащего d измерений. В отличие от метода SANNI, репрезентативный фрагмент метода SAETI может содержать ограниченное количество непересекающихся подпоследовательностей с пропущенными значениями. Предполагается, что количество таких подпоследовательностей не превышает порог $\alpha \cdot (n - m + 1)$, где $0 < \alpha < 1$:

$$|\{\, \boldsymbol{T}_{i,\,m} \in \boldsymbol{S}^m_{\boldsymbol{T}} \mid \exists\, t_j \in T^{(k)}_{i,\,m},\ t_j = \texttt{NaN}\}| \leqslant \alpha \cdot (n - m + 1). \tag{2.15}$$

Типичным значением данного порога является $\alpha = 0.5$. Ограничение вводится для корректного функционирования метода. Метод требует достаточного количества подпоследовательностей для выявления поведенческих шаблонов. Если число подпоследовательностей с пропусками превышает заданный порог, становится невозможно гарантировать достоверность извлеченных шаблонов, что, в свою очередь, снижает точность восстановления.

Описанный выше фрагмент поступает на вход метода SAETI, в результате работы которого будет сформирован набор нейросетевых моделей, способных восстанавливать временной ряд. Предполагается, что после этапа обучения нейросетевые модели, входящие в состав метода, используются для восстановления подпоследовательностей, в которых пропуски могут находиться в любых измерениях любой многомерной точки. Ниже в разделе 2.2.2 представлено описание компонентов метода, включая описание нейросетевых моделей, используемых для восстановления. Процесс обучения нейросетевых моделей, реализующих этапы обработки данных, изложен в разделе 2.2.3. Раздел 2.2.4 посвящен применению обученных моделей для восстановления временных рядов.

2.2.2. Компоненты метода

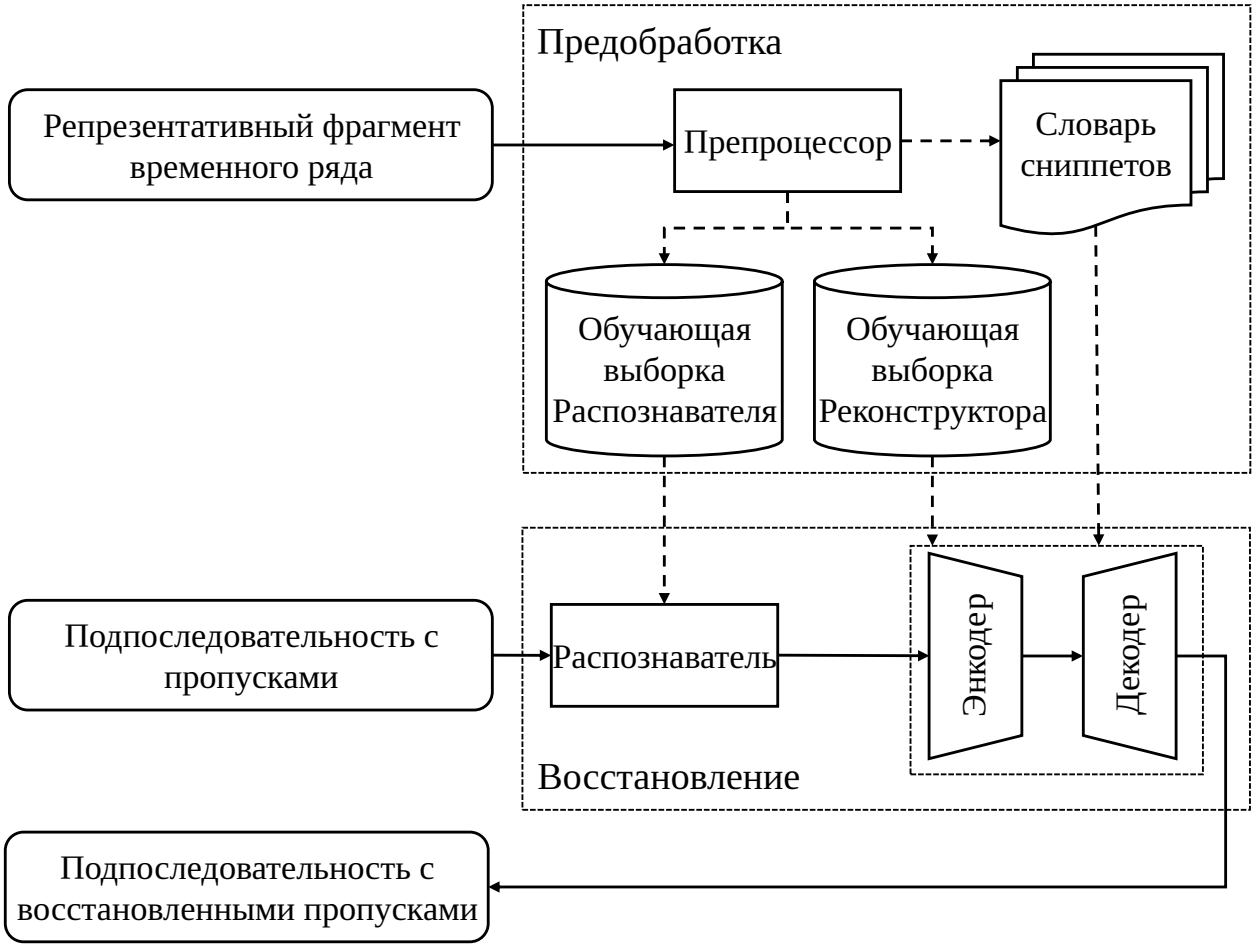


Рис. 2.4. Метод восстановления временного ряда в режиме офлайн

Компоненты предложенного метода представлены на рис. 2.4. Аналогично методу SANNI (см. раздел 2.1.2), предложенный метод включает две основные фазы: обучение и восстановление подпоследовательностей. Фаза обучения модифицирована с учетом того, что репрезентативный фрагмент может содержать пропущенные значения. В связи с этим этап извлечения сниппетов претерпевает изменения. Поиск сниппетов осуществляется только среди подпоследовательностей, не содержащих пропущенных значений. На основе полных подпоследовательностей формируются словарь шаблонов $\boldsymbol{C}_T^m$ и обучающая выборка для нейросетевой модели Распознавателя. В свою очередь, обучающая выборка для нейросетевой модели Реконструктора формируется из всех подпоследовательностей, включая те, которые содержат пропущенные значения.

Во время фазы восстановления каждая подпоследовательность, содержащая пропущенные значения, последовательно обрабатывается нейросетевыми моделями, входящими в состав метода. Результатом их совместного применения является восстановленная подпоследовательность.

Нейросетевая модель Распознаватель

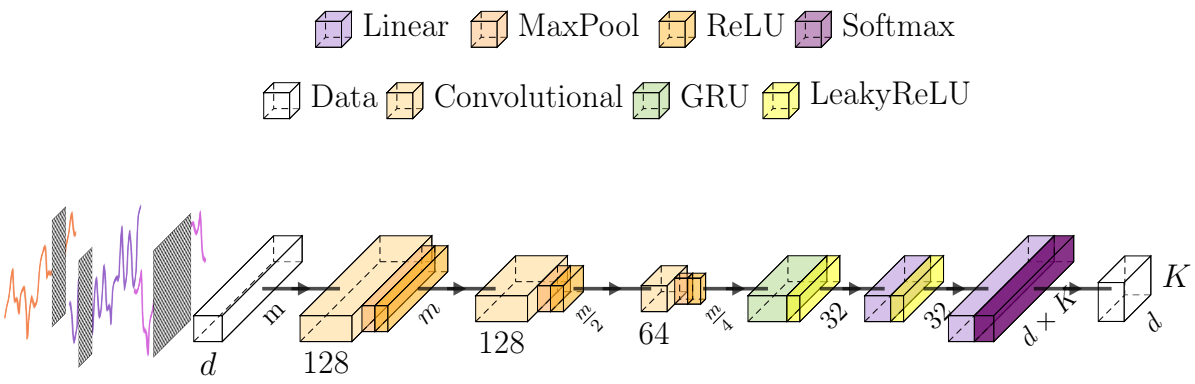


Рис. 2.5. Архитектура нейросетевой модели Распознаватель

На рис. 2.5 представлена архитектура нейронной сети Распознаватель. Распознаватель реализует функцию классификации подпоследовательности (см. формулу 2.5). Данная нейронная сеть принимает на вход многомерную подпоследовательность временного ряда, в которой пропущенные значения заменены на нули. На выходе моделей формируется матрица вероятностей $\boldsymbol{P}$, которая является основой для формирования прогнозного вектора классов $\hat{\boldsymbol{\psi}}$ (см. формулу 2.5).

При сравнении нейросетевой модели Распознавателя в рамках предлагаемого метода и метода SANNI можно выделить несколько ключевых отличий. В предлагаемом методе сверточные слои Распознавателя имеют меньшее количество фильтров при сохранении размера ядра: 128, 128 и 64 соответственно. Дополнительно первый полносвязный слой состоит из 32 нейронов и принимает на вход скрытое состояние, соответствующее последнему временному шагу последовательности.

Изменения в архитектуре модели были внесены по следующим причинам. В связи с изменившимися характеристиками пропущенных дан-

ных (см. формулу 2.15) в подпоследовательностях может присутствовать большее количество пропусков, расположенных случайным образом. После замены на нули, пропуски воспринимаются нейросетевой моделью как шум, что ухудшает способность извлекать и сохранять значимую информацию на каждом временном шаге. Использование только последнего скрытого состояния GRU позволяет агрегировать информацию по всей подпоследовательности, снижая влияние шума и сохраняя наиболее релевантный контекст. Сокращение числа карт признаков и нейронов в полносвязном слое уменьшает риск переобучения и повышает устойчивость модели при работе с зашумленными входными данными.

Нейросетевая модель Реконструктор

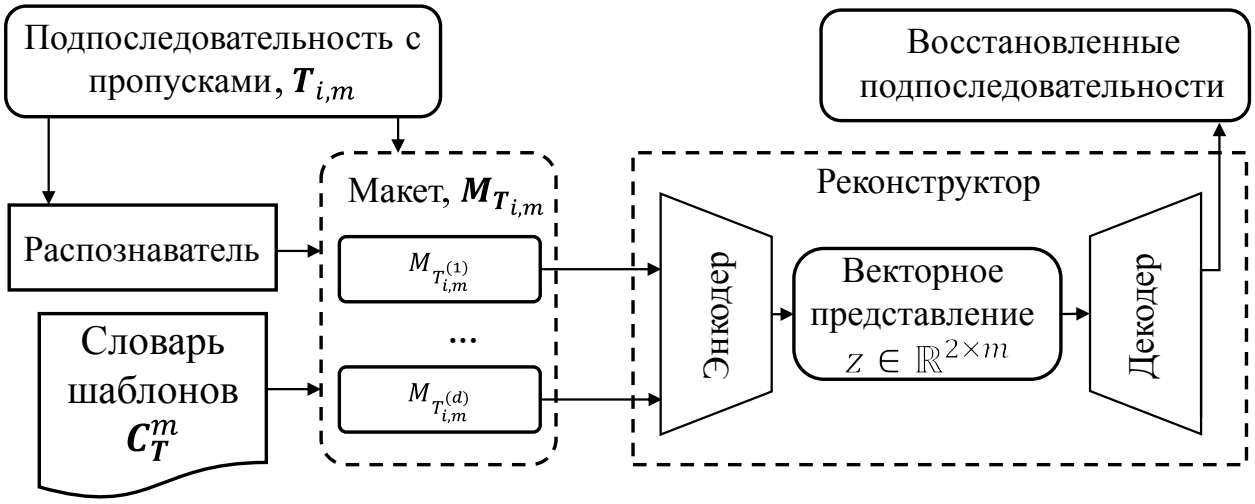


Рис. 2.6. Восстановление пропусков

Рис. 2.6 детализирует восстановление многомерной подпоследовательности. Входная подпоследовательность поступает на вход Распознавателя для формирования шаблонной подпоследовательности $\boldsymbol{F}$ (см. формулу 2.8). На основе шаблонной и входной подпоследовательностей формируется вход нейросетевой модели Реконструктор — макет $\boldsymbol{M}$ (см. формулу 2.9). Реконструктор в данном методе реализуется с помощью модифицированного Автоэнкодера. В общем виде применение Автоэнкодера (см. раздел 1.3.2) для задачи восстановления может быть представлено как композиция следующих функций: кодирование f_{enc} и декодирование f_{dec}.

Для каждой многомерной подпоследовательности $\boldsymbol{T}_{i,m} \in \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$ может быть получена многомерная точка, представляющая собой векторное представление размерности h в пространстве пониженной размерности, содержащие основную информацию о подпоследовательности. Обозначим векторное представление как $z \in \mathbb{R}^h$, множество таких векторов для всех подпоследовательностей ряда обозначим как $Z \in \mathbb{R}^{(n-m+1)\times h}$. Тогда функция кодирования f_{enc} устанавливает соответствия между многомерными подпоследовательностями $\boldsymbol{T}_{i,m}$ и их векторными представлениями z_i. Формально данная функция может быть определена следующим образом:

$$f_{\text{enc}} : \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}} \to Z,\ f_{\text{enc}}(\boldsymbol{T}_{i,m}) = z_i,\ 1 \leqslant i \leqslant n-m+1. \tag{2.16}$$

Декодер выполняет обратное преобразование. Функция декодирования f_{dec} каждому векторному представлению $z \in Z$ сопоставляет восстановленную многомерную подпоследовательность $\overset{\bullet}{\boldsymbol{T}}_{i,\,m} \in \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$:

$$f_{\text{dec}} : Z \to \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}, \quad f_{\text{dec}}(z_i) = \overset{\bullet}{\boldsymbol{T}}_{i,\,m}. \tag{2.17}$$

В задаче восстановления временных рядов предполагается последовательное применение функций кодирования и декодирования. В результате формируется подпоследовательность, в которой не пропущенные значения совпадают с исходными. Пропущенные элементы заменяются приближенными оценками, вычисленными в процессе восстановления:

$$\begin{gathered}
\overset{\bullet}{\boldsymbol{T}}_{i,\,m} = (f_{\text{dec}} \circ f_{\text{enc}})(\boldsymbol{T}_{i,m}), \\
\forall t_j^{(k)} \neq \texttt{NAN} : t_j^{(k)} \approx \overset{\bullet}{t}{}_j^{(k)}, \quad \forall t_j^{k} = \texttt{NAN} : \overset{\bullet}{t}{}_j^{(k)} \neq \texttt{NAN}, \\
t_j^{(k)} \in \boldsymbol{T}_{i,m}, \quad \overset{\bullet}{t}{}_j^{(k)} \in \overset{\bullet}{\boldsymbol{T}}_{i,\,m}, \quad i \leqslant j \leqslant i+m, \quad 1 \leqslant i \leqslant n-m+1.
\end{gathered} \tag{2.18}$$

Из изложенного выше следует, что качество восстановления временных рядов с помощью автоэнкодера определяется степенью, с которой композиция функций кодирования и декодирования аппроксимирует тождественное отображение на подпространстве допустимых последовательностей $\overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$.

Ключевым элементом этой композиции является векторное представление z. Степень соответствия восстановленных подпоследовательностей исходным определяется тем, насколько полно векторное представление z отражает существенную информацию, необходимую для точного восстановления $\boldsymbol{T}_{i,m}$.

Наличие пропусков во входных подпоследовательностях (см. формулу 2.15) приводит к появлению искажений, которые могут существенно повлиять на корректность формируемых представлений. Важно, чтобы Энкодер обладал способностью снижать влияние шума, вызванного неполнотой данных, сохраняя при этом ключевые признаки, необходимые для восстановления структуры временного ряда. Стандартный Энкодер, как правило, восстанавливает пропущенные значения, опираясь на локальные закономерности, присутствующие в наблюдаемых данных. Существует риск того, что при большом количестве пропусков Энкодер будет формировать представления, искаженные вследствие недостатка достоверной информации в наблюдаемых данных. В этом случае нейросетевая модель может опираться на случайные шумовые закономерности, возникающие при инициализации пропусков.

В связи с вышеизложенным предлагается следующая модификация Автоэнкодера. Для повышения качества кодирования восстанавливаемой подпоследовательности на вход Энкодеру дополнительно подается шаблонная подпоследовательность, отражающая ожидаемое поведение процесса. При этом на выходе Декодера формируется восстановленная подпоследовательность. Внесение шаблона во входные данные позволяет Энкодеру учитывать глобальный контекст процесса при анализе наблюдаемых значений. На участках с пропусками результат кодирования определяется не только значениями соседних наблюдений, но и информацией, содержащейся в шаблоне. В результате формируемые векторные представления становятся менее чувствительными к шуму и случайным искажениям. Формально модифицированный Энкодер может быть представлен следующим образом:

$$f_{\text{enc}}(f_{\text{layout}}(\boldsymbol{T}_{i,m}, \boldsymbol{F})) = z_i,\ 1 \leqslant i \leqslant n - m + 1, \tag{2.19}$$

где $f_{\text{layout}} : \mathbb{R}^{m \times d} \times \mathbb{R}^{m \times d} \to \mathbb{R}^{m \times 2d}$ представляет собой функцию объединения многомерной $\boldsymbol{T}_{i,m}$ и шаблонной $\boldsymbol{F}$ подпоследовательностей в макет, согласно формуле 2.9.

Декодер остается стандартным и формирует на выходе восстановленную подпоследовательность. Основной задачей Декодера является восстановление пропущенных элементов исходной подпоследовательности. Если бы выход Декодера включал также шаблонные последовательности, во время обучения декодер вынужденно обучался бы воспроизводить эти шаблоны. В этом случае Энкодер включал информацию о шаблонах в векторное представление, что не является основной целью задачи восстановления. При этом точность воспроизведения шаблонов практически не влияет на качество восстановления исходной подпоследовательности. Добавление шаблонов в выход Декодера не приносит существенной пользы и может усложнить процесс обучения.

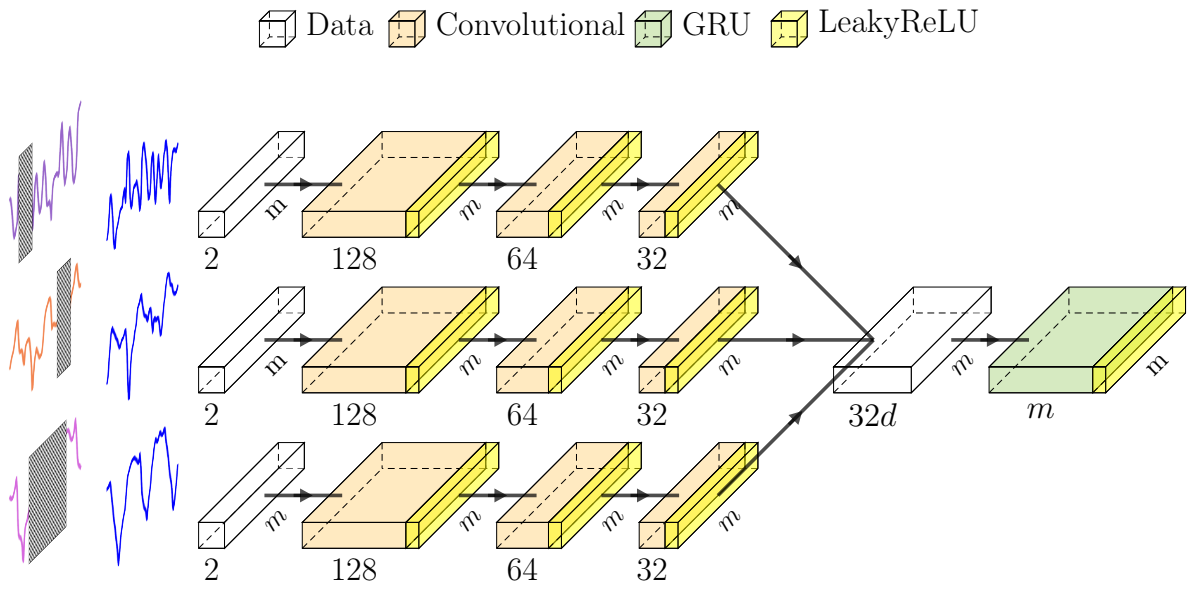


Рис. 2.7. Архитектура нейросетевой модели Энкодер

Рассмотрим более подробно архитектуру нейросетевых моделей, реализующих Энкодер и Декодер. Архитектура Энкодера представлена на рис. 2.7. Энкодер состоит из d серий слоев, каждая из которых включает последовательность сверточных слоев. Каждая последовательность свер-

точных слоев принимает на вход соответствующую матрицу макета. Последовательность слоев состоит из трех сверточных слоев, включающих 128, 64 и 32 фильтра с размером ядра 5. Выходы последовательности сверточных слоев объединяются в матрицу размером $32d \times m$, которая подается на вход рекуррентного слоя.

В отличие от метода SANNI, в предлагаемом решении рекуррентный слой реализуется с использованием двунаправленного GRU-блока с размером скрытого состояния m. Под двунаправленностью в данном случае понимается следующая схема обработки данных. Для входной подпоследовательности формируются векторное представление ее точек. Далее формируются векторные представления для реверс-копии этой последовательности, в которой элементы расположены в обратном порядке. Двунаправленность рекуррентного блока позволяет учитывать как предшествующий, так и последующий контекст для каждого элемента подпоследовательности. Поскольку пропуски могут возникать не только в начале или конце, но и в центральной части подпоследовательности, использование информации, расположенной вокруг пропуска, позволит повысить точность восстановления. Рекуррентный слой агрегирует признаки, извлеченные сверточными слоями, и формирует векторное представление подпоследовательности ввиде двустрочной матрицы, где первая строка соответствует векторному представлению последней точки исходной подпоследовательности. Вторая строка соотвествует векторному представлению последней точки реверс-копии подпоследовательности.

Векторное представление, полученное на выходе Энкодера, поступает на вход декодера (см. рис. 2.8). Архитектура Декодера является зеркальной копией структуры Энкодера, в которой каждый сверточный слой заменен на соответствующий транспонированный сверточный слой. *Транспонированный сверточный слой (transposed convolutional layer)* [122] используется для увеличения пространственного разрешения входных данных. Транспонированная свертка представляет собой операцию, противоположную обычной свертке. В результате выполнения данной операции увеличивается размерность данных за счет вставки нулевых значений между ис-

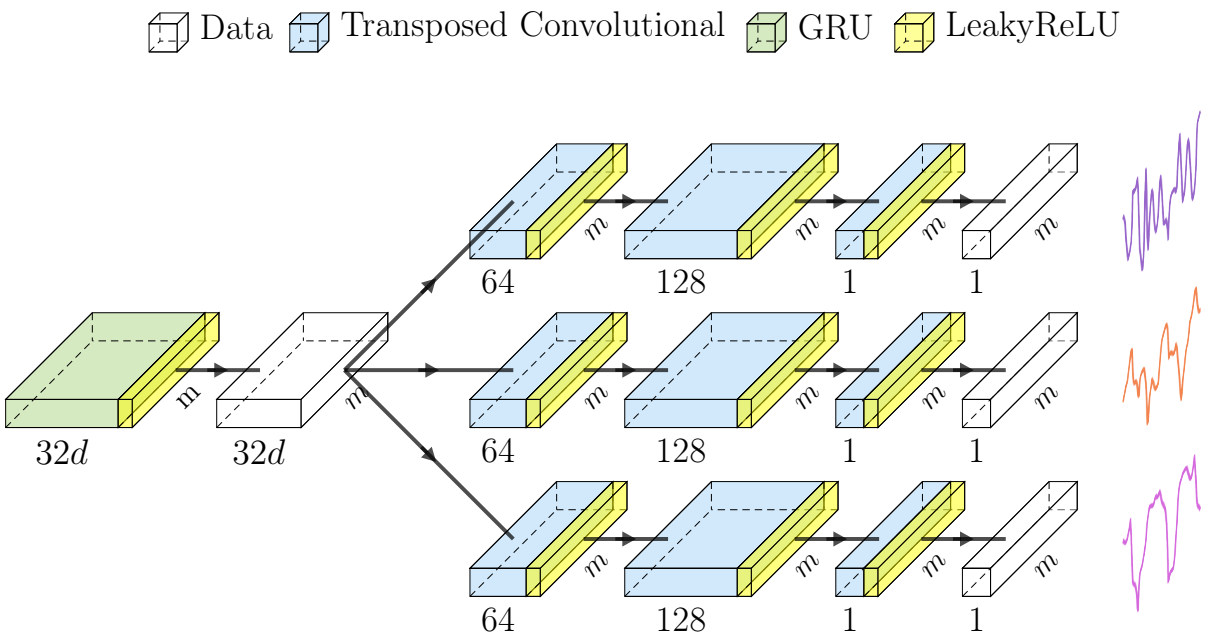


Рис. 2.8. Архитектура нейросетевой модели Декодер

ходными входными значениями. После этого к получившимся данным применяется свертка, что приводит к увеличению пространственного разрешения выходных данных. Результатом применения серий транспонированных сверточных слоев является восстановленная подпоследовательность $\overset{\bullet}{\boldsymbol{T}}_{i,m}$.

2.2.3. Обучение нейросетевых моделей

Обучение нейросетевых моделей, реализующих восстановление временных рядов, выполняется аналогично этапу обучения из метода SANNI (см. раздел 2.1.3). В методе SAETI данный этап модифицируется следующим образом: вносятся изменения на стадии поиска шаблонов, формирования обучающих выборок модели, а также в функции потерь Реконструктора.

Модификация поиска шаблонов

В предлагаемом методе используется модифицированный алгоритм поиска шаблонов, отличающийся от оригинального алгоритма, описанного в [66]. Основное отличие заключается в предварительной обработке временного ряда и организации вычислений. В частности, исходный времен-

ной ряд T разбивается на множество пересекающихся подпоследовательностей фиксированной длины m. Из полученного множества исключаются подпоследовательности обладающие следующими свойствами: подпоследовательности, содержащие хотя бы одно пропущенное значение, и подпоследовательности, для которых доля перекрытия с другими подпоследовательностями превышает порог, равный 50% точек.

Для каждой оставшейся подпоследовательности рассчитывается матричный профиль относительно остальных подпоследовательностей в получившемся множестве. После получения всех профилей выполняется поиск шаблонов в соответствии с исходным алгоритмом. В результате формируются словарь шаблонов $\boldsymbol{C}_T^m$ и множество векторов классов для всех подпоследовательностей.

Формирование обучающих выборок

Для формального описания формирования обучающих выборок метода SAETI введем две функции: замены пропусков f_{replace} и добавления пропусков f_{zero}. Функция замены пропусков производит замену всех пропусков в подпоследовательности из множества $\overset{\bullet}{\boldsymbol{S}}{}_{\boldsymbol{T}}^{m}$ на константу c. Формально данная функция может быть представлена следующим образом:

$$\begin{gathered} f_{\text{replace}} : \overset{\bullet}{\boldsymbol{S}}{}_{\boldsymbol{T}}^{m}, \mathbb{R} \to \overset{\bullet}{\boldsymbol{S}}{}_{\boldsymbol{T}}^{m}, \quad f_{\text{replace}}(\boldsymbol{T}_{i,\,m}, c) = \overset{\bullet}{\boldsymbol{T}}_{i,\,m}, \\ \forall t_j^{(k)} = \texttt{NAN} \quad \overset{\bullet}{t}{}_j^{(k)} = c, \quad c \in \mathbb{R}. \end{gathered} \tag{2.20}$$

Функция добавления пропусков f_{zero} реализует внедрение искусственных пропусков в подпоследовательности из множества $\overset{\bullet}{\boldsymbol{S}}{}_{\boldsymbol{T}}^{m}$ путем замены части значений на 0. Выбор элементов для замены осуществляется равновероятно. Общее количество пропусков не превышает порог $\lfloor p \cdot m \cdot d \rfloor$, $p \in [0, 1]$. Функция f_{zero} необходима для имитации пропусков в обучающих данных. Случайная замена части элементов на нули с контролируемой долей пропусков p обеспечивает обучение нейросетевой модели обработке ситуаций, приближенных к реальным условиям. Формально данная функция

может быть представлена следующим образом:

$$\begin{gathered} f_{\text{zero}} : \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}, [0,1] \to \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}, f_{\text{zero}}(\boldsymbol{T}_{i,m}, p) = \overset{\bullet}{\boldsymbol{T}}_{i,m}, \\ \mid \{\overset{\bullet}{t}{}^{(k)}_{j} \mid \overset{\bullet}{t}{}^{(k)}_{j} = 0, \quad t^{(k)}_{j} \neq 0\} \mid = \lfloor p \cdot m \cdot d \rfloor, \quad p \in [0,1]. \end{gathered} \tag{2.21}$$

В качестве входных данных в обучающей выборке Распознавателя полагаются многомерные подпоследовательности из множества $\boldsymbol{S}^{m}_{\boldsymbol{T}}$, для которых на этапе поиска шаблонов была получена разметка классов. Перед подачей на вход нейронной сети к подпоследовательностям применяется функция добавления пропусков f_{zero} с параметром p. Выходными значениями полагаются вектора классов $\boldsymbol{\psi}$, полученные для подпоследовательностей на этапе предварительной обработки. Формальное определение обучающей выборки Распознавателя выглядит следующим образом:

$$\begin{aligned} D_{\text{Recognizer}} = \{\langle \boldsymbol{X},\, \boldsymbol{Y}\rangle \;\mid\; & \boldsymbol{X} = f_{\text{zero}}(\boldsymbol{T}_{i,m}, p),\; \boldsymbol{T}_{i,m} \in \boldsymbol{S}^{m}_{\boldsymbol{T}} \\ & \boldsymbol{Y} = \boldsymbol{\psi}_i,\; \boldsymbol{\psi}_i = \boldsymbol{f}_{\text{class}}(\boldsymbol{T}_{i,m}), \quad p = [0,1], \\ & 1 \leqslant i \leqslant n - m + 1,\; 1 \leqslant k \leqslant d,\; 1 \leqslant s \leqslant K\}. \end{aligned} \tag{2.22}$$

Входными данными обучающей выборки Реконструктора считаются все подпоследовательности временного ряда из множества $\overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$. Поскольку в этом множестве могут присутствовать подпоследовательности с пропущенными значениями, на первом этапе к ним применяется функция замены пропусков f_{replace} с параметром $c = 0$. Далее к полученным полным подпоследовательностям применяется функция добавления пропусков f_{zero} с заданным параметром p. В качестве выходных данных используются исходные подпоследовательности. Формально обучающая выборка Реконструктора определяется следующим образом:

$$\begin{aligned} D_{\text{Reconstructor}} = \{\langle \boldsymbol{X},\, \boldsymbol{Y}\rangle \;\mid\; & \boldsymbol{X} = f_{\text{zero}}(f_{\text{replace}}(\boldsymbol{T}_{i,m}, c), p), \\ & \boldsymbol{Y} = \boldsymbol{T}_{i,m},\; p = [0,1],\; c \in \mathbb{R} \\ & 1 \leqslant i \leqslant n - m + 1,\; 1 \leqslant k \leqslant d\}. \end{aligned} \tag{2.23}$$

Поскольку обучающая выборка содержит подпоследовательности, для которых на этапе поиска шаблонов не были сформированы векторы классов, обученный Распознаватель используется для обучения Реконструктора.

Вычисление ошибки

Функция потерь Распознавателя соответствует функции потерь, используемой в методе SANNI (см. формулу 2.13). Ввиду того, что архитектура Автоэнкодера предусматривает не только восстановление пропусков в подпоследовательности, но и ее кодирование с последующим восстановлением (см. формулу 2.18), функция потерь должна отражать качество реконструкции исходной подпоследовательности из ее векторного представления. Следовательно, при вычислении функции потерь учитываются значения всех элементов входной подпоследовательности.

Однако специфика задачи восстановления подпоследовательностей с пропущенными значениями накладывает определенные ограничения на процесс обучения. На практике реальные значения элементов временного ряда могут быть известны не для всех многомерных точек, что приводит к необходимости модификации функции потерь. Во избежание искажения оценки качества восстановления, при вычислении функции потерь учитываются только те элементы, для которых доступны истинные значения. Таким образом функция потерь Реконструктора $L_{\text{Reconstructor}} : \mathbb{R}^{m \times d} \times \mathbb{R}^{m \times d} \to \mathbb{R}$ может быть представлена следующим образом:

$$
\begin{gathered}
L_{\text{Reconstructor}}(\overset{\bullet}{\boldsymbol{T}}_{i,m},\, \boldsymbol{Y}) = \frac{1}{l} \sum_{i=1}^{d} \sum_{j=1}^{m} \sqrt{1_{\{\texttt{NAN}\}}(Y_j^{(i)}) \cdot (Y_j^{(i)} - \overset{\bullet}{t}_j^{(i)})^2}, \\
l = |\{Y_j^{(i)} \mid \forall Y_j^{(i)} \neq \texttt{NAN}\}|, \quad 1_{\{\texttt{NAN}\}}(x) = \begin{cases} 1, & x \neq \texttt{NAN}, \\ 0, & x = \texttt{NAN}. \end{cases}
\end{gathered}
\tag{2.24}
$$

2.2.4. Применение метода

На этапе восстановления из исходного ряда извлекаются индексы многомерных временных точек, содержащих хотя бы одно пропущенное значение. Для каждой i-й пропущенной точки в окрестности $[i - m, i + m]$ выбирается подпоследовательность, содержащая минимально возможное количество пропусков, при этом пропущенное значение располагается как можно ближе к центру выбранного интервала. Размещение пропущенного значения вблизи центра подпоследовательности особенно целесообразно при использовании двунаправленных рекуррентных слоев, поскольку обеспечивает модели доступ к максимально возможному количеству полных наблюдений.

Выбранные подпоследовательности нормализуются и подаются на вход Распознавателю, который для каждой из них формирует соответствующую шаблонную подпоследовательность. Далее нормализованные и шаблонные последовательности поступают на вход Реконструктора, формирующего восстановленную версию подпоследовательности. При формировании окончательного результата восстановления учитываются исключительно те элементы подпоследовательности, которые отсутствовали в исходных данных.

2.3. Выводы по главе 2

В данной главе представлены два нейросетевых метода восстановления потоковых данных, представленных в форме временных рядов. Методы предназначены для различных сценариев обработки данных. Предложенные методы основаны на концепции применения шаблонов активностей и нейросетевых технологий для повышения точности восстановления.

Предложен нейросетевой метод восстановления потоковых данных, представленных в форме временных рядов, в режиме онлайн, который назван SANNI (Snippet and Artificial Neural Network-based Imputation). Для повышения точности восстановления в данном методе используются шаб-

лоны (подпоследовательности временных рядов, отражающих типичные действия субъекта). Восстановление реализуется с использованием двух последовательно применяемых нейросетевых моделей: Распознавателя и Реконструктора. Распознавателя получает на вход подпоследовательность временного ряда с пропусками и формирует вектор меток, содержащий наиболее вероятные классы активности для каждого измерения. На основе вектора формируется шаблонная подпоследовательность, которая вместе с входной последовательностью поступает на вход Реконструктора. Реконструктор, в свою очередь, восстанавливает пропущенные значения. Метод SANNI представляет собой инструмент онлайн-восстановления пропусков во временных рядах, предназначенный для использования на этап применения нейросетевых моделей и обеспечивающий восстановление текущих пропущенных значений на основе предыдущих наблюдений.

Предложен новый метод восстановления потоковых данных, представленных в форме временных рядов, в режиме офлайн, который был назван SAETI (Snippet based Autoencoder for Timeseries Imputation). SAETI развивает метод SANNI, адаптируя его под особенности офлайн-восстановления временных рядов. Распознаватель метода SAETI упрощен для повышения устойчивости к шуму. Для реализации Реконструктора метода SAETI используется Автоэнкодера, Энкодер которого получает поведенческие шаблоны в качестве дополнительного входа. На выходе Реконструктора формируется восстановленная подпоследовательность временного ряда, в которой все пропуски заменены синтетическими значениями. Метод SAETI предназначен для использования на этапе предварительной обработки данных в жизненном цикле нейросетевых моделей обработки временных рядов. Он обеспечивает корректное заполнение пропусков в исходных временных рядах, формируя полные и качественные обучающие выборки для последующих этапов жизненного цикла.

Результаты, приведенные в этой главе, опубликованы в статьях [8, 9, 161].

Глава 3. Оценка точности нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов

В данной главе предложены новые методы анализа нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, которые могут использоваться в качестве инструментальных средств разработки различных этапов их жизненного цикла. В разделе 3.1 предложена новая функция потерь MPDE, предназначенная для обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Раздел 3.2 посвящен новому методу tsGAP, предназначенному для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Предложенный метод может применяться в качестве инструментального средства предварительной оценки на этапе выбора целевой модели.

3.1. Функция потерь для обучения нейросетевых моделей восстановления

В данном разделе описана новая функция потерь для обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, названная MPDE (Mean Profile Distance Error). Предложенная функция потерь направлена на повышение точности восстановления за счет учета как отдельно взятых точек подпоследовательностей, так и нормализованного расстояния между окнами подпоследовательностей. Для интеграции MPDE в существующие фреймворки глубокого обучения предложен параллельный алгоритм ее вычисления. Алгоритм разработан с учетом особенностей фреймворка PyTorch и позволяет производить обучение моделей с использованием MPDE без существенных накладных расходов.

3.1.1. Формальное определение функции потерь MPDE

Информацию, содержащуюся во временных рядах, можно декомпозировать различными способами. Например, временной ряд можно разделить на две взаимосвязанные составляющие [79]. Первая отражает основные численные показатели ряда, включая среднее значение, амплитуду, экстремальные значения и др. Вторая содержит поведенческие особенности подпоследовательностей и закономерности, важные для анализа поведения системы, выявления аномалий и оценки повторяемости паттернов. Для подтверждения вышеизложенного рассмотрим следующие примеры.

В различных приложениях Интернета вещей (IoT) поведенческие характеристики временных рядов играют важную роль в выявлении закономерностей и аномалий. Анализ среднего значения ряда данных с датчиков вибрации промышленного оборудования позволяет определить, функционирует ли оборудование в штатном режиме. Однако при диагностике состояния оборудования не менее важным является сохранение и анализ формы ряда. Дефекты подшипников нередко вызывают ударные вибрации, которые формируют регулярные характерные пики на огибающем спектре [34].

В физиологических временных рядах структурные характеристики играют ключевую роль в диагностике и анализе состояния пациента. Значения ряда электрокардиограммы (ЭКГ) отражают электрическое напряжение сердца в каждый момент времени, измеряемое в милливольтах (мВ), и используются кардиологами для оценки общего уровня активности сердца. Не менее значимым является анализ формы временного ряда, отражающей динамику электрической активности сердца. Зубец P, комплекс QRS и зубец T отражают электрическую активность сердца и формируют характерный сердечный ритм, который служит основой для количественной оценки физиологического состояния пациента. Например, анализ интервала QT позволяет выявлять изменения электрофизиологии миокарда и оценивать риск аритмий [17].

Учитывая вышеизложенное, при восстановлении временных рядов важно не только минимизировать ошибки между отдельными точками, но и со-

хранять локальные структурные закономерности подпоследовательностей, отражающие поведенческие особенности ряда. В этом разделе представлено формальное определение функции потерь MPDE, предложенной для обучения нейросетевых моделей восстановления временных рядов. Функция MPDE при сравнении подпоследовательностей учитывает отклонения между отдельными точками подпоследовательностей и их поведенческое сходство. На вход функции поступают две подпоследовательности длиной m: $\boldsymbol{X}$, представляющая собой восстановленные нейросетевой моделью данные, и $\boldsymbol{Y}$, содержащая истинные значения временного ряда.

Функция потерь MPDE

Рассмотрим формальное определение функции MPDE для одномерного ряда, то есть для случая $X, Y \in S_T^m$, после чего обобщим его на многомерный случай. Функция MPDE определяется как среднее значение расстояний между парами окон сравниваемых подпоследовательностей. Каждое окно имеет длину ℓ, где $\ell < m$, и предполагается, что длина окна пропорциональна длине подпоследовательности m. Эта пропорция задается параметром γ, удовлетворяющим условию $0 < \gamma < 1$, так что $\ell = \lfloor \gamma m \rfloor$. Общее число окон обозначим как c, где $c = m - \ell + 1$. При этом сравнение проводится между окнами с одинаковыми индексами.

Чтобы модель во время обучения учитывала обе рассмотренные выше составляющие информации временного ряда, используется функция потерь, представленная в виде суммы двух слагаемых. Первое слагаемое представляет собой евклидово расстояние и оценивает различие между отдельными точками подпоследовательностей. Второе слагаемое вычисляется на основе z-нормализованных окон и оценивает поведенческое сходство подпоследовательностей. Для вычисления расстояний между окнами подпоследовательностей используются евклидово расстояние и его z-нормализованная версия, обозначаемые как $\mathrm{ED}(\cdot,\cdot)$ и $\widehat{\mathrm{ED}}(\cdot,\cdot)$ соответственно. Для вычисления евклидовых расстояний под корень вносится машинный эпсилон $\varepsilon > 0$, который представляет собой минимальную разницу между 1 и ближайшим большим числом, представимым в формате с пла-

вающей запятой, и обеспечивает численную устойчивость градиентов во время обратного распространения ошибки:

$$\mathrm{ED}(X,Y) = \sqrt{\sum_{i=1}^{\ell}(x_i - y_i)^2 + \varepsilon}, \quad \widehat{\mathrm{ED}}(X,Y) = \sqrt{\sum_{i=1}^{\ell}(\hat{x}_i - \hat{y}_i)^2 + \varepsilon}, \quad (3.1)$$

где z-нормализация представляет собой следующее:

$$\widehat{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \mu_x = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i, \quad \sigma_x = \sqrt{\frac{1}{\ell - 1}\sum_{i=1}^{\ell}(x_i - \mu_x)^2 + \varepsilon}. \quad (3.2)$$

Веса указанных слагаемых, α и β соответственно, представляют собой вещественные числа из отрезка (0,1) и являются параметрами функции потерь. Функция потерь MPDE вычисляется следующим образом:

$$\mathrm{MPDE}(X,\, Y) = \frac{1}{2 \cdot c}\sum_{i=1}^{c}\Big(\alpha \mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) + \beta\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell})\Big), \quad 0 < \alpha,\ \beta < 1. \quad (3.3)$$

Свойства функции потерь

К основным свойствам функций потерь при обучении нейросетевых моделей относят следующие свойства (см. обзоры [86, 125]): неотрицательность, наличие глобального минимума и дифференцируемость. *Неотрицательность* функции потерь обеспечивает количественно интерпретируемую шкалу оценки качества модели: низкие значения соответствуют прогнозам, близким к истинным значениям целевой переменной, тогда как увеличение значения функции потерь отражает рост расхождения между прогнозом и наблюдаемыми данными. Существование *глобального минимума* функции потерь является важным условием, так как оно определяет конечную цель обучения — достижение точки, в которой функция принимает минимальное значение, соответствующее совпадению предсказанных и истинных значений. *Дифференцируемость* функции потерь является необходимым условием для применения градиентных методов оптимизации. Это

свойство обеспечивает возможность вычисления направления и величины обновления весов нейросетевой модели на каждой итерации обучения.

Следует отметить, что выпуклость функции потерь является желательным, но не обязательным свойством. Многие современные функции потерь являются невыпуклыми и при этом демонстрируют успешное применение на практике. Примерами таких функций являются Soft-DTW [38], RMSE, Huber Loss и Log-Cosh Loss [64, 112]. Невыпуклые функции потерь могут эффективно использоваться для обучения нейросетевых моделей, в сочетании с градиентными или стохастическими методами оптимизации [90]. Кроме того, исследования демонстрируют, что для широкого класса гладких невыпуклых функций стохастический градиентный спуск, используемый для обучения нейросетевых моделей, способен сходиться к стационарным точкам, которые зачастую оказываются близкими к глобальному минимуму [148]. Таким образом, невыпуклость функции потерь не является препятствием для ее применения на практике.

Предложенная в настоящем исследовании функция потерь MPDE является строго положительной, дифференцируемой и обладает глобальным минимумом. Доказательства соответствующих утверждений приводятся ниже.

Лемма 1. Пусть $p \geqslant 0$ и $\gamma > 0$. Тогда $\sqrt{p+\gamma} \geqslant \sqrt{\gamma}$, причем $\sqrt{p+\gamma} \geqslant \sqrt{\gamma}$, выполняется тогда и только тогда, когда $p = 0$.

Доказательство. Так как $p \geqslant 0$, имеем $p + \gamma \geqslant \gamma$. Поскольку функция $\sqrt{\cdot}[0,\infty)$, из неравенства $p + \gamma \geqslant \gamma$ следует

$$\sqrt{p+\gamma} \geqslant \sqrt{\gamma}. \tag{3.4}$$

Если при этом $\sqrt{p+\gamma} = \sqrt{\gamma}$, в силу монотонности квадратного корня имеем $p + \gamma = \gamma$, откуда $p = 0$. Обратно, при $p = 0$ равенство очевидно. □

Утверждение 1. ***Положительность.*** *Пусть* $X, Y \in \mathbb{R}^m$, $0 < \alpha, \beta < 1$, $\ell < m, c = m - \ell + 1$. *Тогда* $\mathrm{MPDE}(X, Y) \geqslant \sqrt{\varepsilon}$.

Доказательство. Для каждого окна $X_{i,\ell}$ и $Y_{i,\ell}$ положим

$$a_i = \sum_{j=1}^{\ell} (x_{i+j} - y_{i+j})^2. \tag{3.5}$$

Заметим, что каждый член суммы $(x_{i+j} - y_{i+j})^2 \geqslant 0$, поскольку квадрат любого действительного числа неотрицателен. Следовательно, сумма конечного числа неотрицательных чисел также неотрицательна:

$$a_i \geqslant 0. \tag{3.6}$$

По Лемме 1 имеем

$$\mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) = \sqrt{a_i + \varepsilon} \geqslant \sqrt{\varepsilon}. \tag{3.7}$$

Аналогично для нормализованной версии $\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell})$ положим

$$\hat{a}_i = \sum_{j=1}^{\ell} (\hat{x}_{i+j} - \hat{y}_{i+j})^2. \tag{3.8}$$

Поскольку квадрат любого действительного числа неотрицателен, получаем

$$\hat{a}_i = \sum_{j=1}^{\ell} (\hat{x}_{i+j} - \hat{y}_{i+j})^2 \geqslant 0. \tag{3.9}$$

Применяя Лемму 1, получаем

$$\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell}) = \sqrt{\hat{a}_i + \varepsilon} \geqslant \sqrt{\varepsilon}. \tag{3.10}$$

Так как для каждого окна $i = 1, \ldots, c$ выполнено

$$\mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) \geqslant \sqrt{\varepsilon}, \quad \widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell}) \geqslant \sqrt{\varepsilon}, \tag{3.11}$$

и коэффициенты $\alpha, \beta > 0$, получаем, что каждое слагаемое (см. формулу 3.3)

$$\alpha \operatorname{ED}(X_{i,\ell}, Y_{i,\ell}) + \beta \widehat{\operatorname{ED}}(X_{i,\ell}, Y_{i,\ell}) \geqslant \sqrt{\varepsilon}. \tag{3.12}$$

Следовательно, усреднение по всем c окнам также строго положительно:

$$\operatorname{MPDE}(X, Y) = \frac{1}{2c} \sum_{i=1}^{c} \Big(\alpha \operatorname{ED}(X_{i,\ell}, Y_{i,\ell}) + \beta \widehat{\operatorname{ED}}(X_{i,\ell}, Y_{i,\ell}) \Big) \geqslant \sqrt{\varepsilon}. \tag{3.13}$$

□

Утверждение 2. ***Наличие глобального минимума.*** *Пусть* $X, Y \in \mathbb{R}^m$ $0 < \alpha, \beta < 1$, $\ell < m$, $c = m - \ell + 1$. *Тогда минимум функции* $\operatorname{MPDE}(\cdot\,,\cdot)$ *достигается при* $X = Y$.

Доказательство. Доказательство существования глобального минимума функции MPDE проводится в три шага. *На первом шаге* рассматривается функция $\operatorname{ED}(X, Y)$ и находится ее глобальный минимум. *На втором шаге* анализируется функция $\widehat{\operatorname{ED}}(X, Y)$ и описывается множество ее минимумов. *На третьем шаге* рассматривается функция $\operatorname{MPDE}(\cdot\,,\cdot)$, которая включает как $\operatorname{ED}(\cdot\,,\cdot)$, так и $\widehat{\operatorname{ED}}(\cdot\,,\cdot)$, и доказывается, что добавление обычного евклидова расстояния делает глобальный минимум функции уникальным, достигаемым только при точном совпадении всех окон.

Шаг 1. Глобальный минимум функции $\operatorname{ED}(\cdot\,,\cdot)$

Частные производные $\operatorname{ED}(\cdot\,,\cdot)$ по x_k, $i \leqslant k \leqslant i + \ell$ окна $X_{i,\ell}$, имеют вид:

$$\frac{\partial \operatorname{ED}(X_{i,\ell}, Y_{i,\ell})}{\partial x_k} = \frac{x_k - y_k}{\sqrt{\sum_{j=i}^{i+\ell} (x_j - y_j)^2 + \varepsilon}}. \tag{3.14}$$

Для определения точек экстремума функции $\operatorname{ED}(\cdot\,,\cdot)$ по элементам окна $X_{i,\ell}$, приравняем ее частные производные к нулю:

$$\frac{\partial \operatorname{ED}(X_{i,\ell}, Y_{i,\ell})}{\partial x_k} = \frac{x_k - y_k}{\sqrt{\sum_{j=i}^{i+\ell} (x_j - y_j)^2 + \varepsilon}} = 0. \tag{3.15}$$

Поскольку знаменатель в формуле 3.15 строго положителен (см. формулу 3.7) для всех $X_{i,\ell}$, равенство производной нулю может выполняться тогда и только тогда, когда числитель равен нулю, то есть:

$$x_k - y_k = 0 \quad \Longleftrightarrow \quad x_k = y_k. \tag{3.16}$$

Таким образом, единственная точка экстремума функции $\mathrm{ED}(\cdot\,,\cdot)$ возникает исключительно при совпадении всех точек сравниваемых окон. Поскольку функция ограничена снизу значением $\sqrt{\varepsilon}$ (см. формулу 3.7), равенство $\mathrm{ED}(X,Y) = \sqrt{\varepsilon}$ достигается тогда при $X = Y$. Следовательно, точка $X = Y$ является глобальным минимумом функции $\mathrm{ED}(X,Y)$.

Шаг 2. Минимум функции $\widehat{\mathrm{ED}}(\cdot\,,\cdot)$

Аналогично, рассмотрим функцию $\widehat{\mathrm{ED}}(X,Y)$. Минимум этой функции достигается в случае, если $\widehat{X} = \widehat{Y}$. Из определения нормализации (формула 3.2) следует:

$$\frac{X - \mu_x \mathbf{1}}{\sigma_x} = \frac{Y - \mu_y \mathbf{1}}{\sigma_y}, \tag{3.17}$$

где $\mathbf{1} = \{1, 1, ..., 1\}^\intercal$ — вектор, состоящий из ℓ единиц.

Умножив обе стороны формулы 3.17 на $\sigma_y > 0$, получаем:

$$Y - \mu_y \mathbf{1} = \frac{\sigma_y}{\sigma_x}(X - \mu_x \mathbf{1}) = \lambda(X - \mu_x \mathbf{1}), \quad \frac{\sigma_y}{\sigma_x} = \lambda, \quad \lambda > 0. \tag{3.18}$$

Подставляя выражение $Y - \mu_y \mathbf{1} = \lambda(X - \mu_x \mathbf{1})$ в определение нормализованного вектора (см. формулу 3.17), имеем:

$$\hat{Y} = \frac{Y - \mu_Y \mathbf{1}}{\sigma_Y} = \frac{\lambda(X - \mu_X \mathbf{1})}{\sigma_Y} = \frac{\lambda(X - \mu_X \mathbf{1})}{\lambda\,\sigma_X} = \frac{X - \mu_X \mathbf{1}}{\sigma_X} = \hat{X}. \tag{3.19}$$

Следовательно, глобальный минимум функции $\widehat{\mathrm{ED}}(X,Y)$ достигается для всех пар (X,Y), которые имеют пропорциональные параметры нормализации (μ, σ) и одинаковые значения после нормализации. Любые такие пары дают одинаковое минимальное значение, поэтому минимум не уникален.

Шаг 3. Глобальный минимум функции MPDE

Если для некоторого i-го окна выполняется

$$\hat{X}_{i,\ell} = \hat{Y}_{i,\ell}, \quad X_{i,\ell} \neq Y_{i,\ell}, \tag{3.20}$$

то

$$\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell}) = \sqrt{\varepsilon}, \tag{3.21}$$

но при этом

$$\mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) = \sqrt{\sum_{j=i}^{i+\ell}(x_j - y_j)^2 + \varepsilon} > \sqrt{\varepsilon}, \tag{3.22}$$

так как существуют элементы окна, такие что $x_j \neq y_j$.

Таким образом, несмотря на то, что существует множество пар окон, для которых выполняется $\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell}) = \sqrt{\varepsilon}$, вклад $\mathrm{ED}(\cdot\,,\cdot)$ исключает возможность достижения глобального минимума при $X_{i,\ell} \neq Y_{i,\ell}$. Следовательно, глобальный минимум MPDE достигается только при $X = Y$. □

Утверждение 3. ***Дифференцируемость.*** *Рассмотрим нейросетевую модель, порождающую выходной вектор*

$$X = F(Z, \theta) \in \mathbb{R}^m, \tag{3.23}$$

где θ обозначает совокупность всех весов нейросетевой модели, а Z входной вектор модели. Пусть целевой вектор $Y \in \mathbb{R}^m$ фиксирован, а функция потерь имеет вид

$$L(X, Y) = \mathrm{MPDE}(X, Y), \tag{3.24}$$

где $\ell < m$, $c = m - \ell + 1$, а $\varepsilon > 0$, $0 < \alpha, \beta < 1$. Градиент функции потерь MPDE *по выходам последнего слоя нейросети вычисляется как сумма вкладов всех окон, содержащих соответствующую компоненту.*

Для компоненты x_k выходного вектора X получаем

$$\frac{\partial \mathrm{MPDE}}{\partial x_k} = \frac{1}{2c} \sum_{i=i_{min}}^{i_{max}} \left(\alpha \frac{x_k - y_k}{\mathrm{ED}(X_{i,\ell}, Y_{i,\ell})} + \right.$$
$$\left. + \beta \frac{1}{\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell})} \sum_{j=1}^{\ell} (\hat{x}_{i+j} - \hat{y}_{i+j}) \frac{\partial \hat{x}_{i+j}}{\partial x_k} \right). \tag{3.25}$$
$$i_{\min} = \max(1, k - \ell), i_{\max} = \min(k, m - \ell + 1).$$

где частная производная нормализации по элементу окна x_k равна

$$\frac{\partial \hat{x}_j^{(i)}}{\partial x_k} = \frac{\delta_{jk} - \frac{1}{\ell}}{\sigma_{X_{i,\ell}}} - \frac{x_j^{(i)} - \mu_{X_{i,\ell}}}{(\sigma_{X_{i,\ell}})^2} \cdot \frac{x_k^{(i)} - \mu_{X_{i,\ell}}}{(\ell - 1)\, \sigma_{X_{i,\ell}}}. \tag{3.26}$$

где δ_{jk} — символ Кронекера, определяемый следующим образом:

$$\delta_{jk} = \begin{cases} 1, & \text{если } j = k, \\ 0, & \text{если } j \neq k. \end{cases} \tag{3.27}$$

Доказательство. Поскольку подкоренная сумма в определении $\mathrm{ED}(\cdot\,,\cdot)$ содержит добавку $\varepsilon > 0$, выражения под корнем строго положительны 3.7; стандартные правила дифференцирования композиции и суммы применимы, поэтому все функции гладкие и частные производные по компонентам x_k существуют.

Для i-го окна и для компоненты x_k, $1 \leqslant k \leqslant m$ имеем

$$\frac{\partial}{\partial x_k} \mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) = \frac{\partial}{\partial x_k} \sqrt{S^{(i)} + \varepsilon} = \frac{1}{2\mathrm{ED}(X_{i,\ell}, Y_{i,\ell})} \cdot \frac{\partial S^{(i)}}{\partial x_k}, \tag{3.28}$$

где $S^{(i)} = \sum_{j=1}^{\ell} (x_{i+j} - y_{i+j})^2$. Поскольку $\partial S^{(i)} / \partial x_k = 2(x_k - y_k)$, получаем

$$\frac{\partial}{\partial x_k} \mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) = \frac{x_k - y_k}{\mathrm{ED}(X_{i,\ell}, Y_{i,\ell})}. \tag{3.29}$$

Заметим, что

$$\frac{\partial}{\partial x_k}\sum_{r=i}^{i+\ell}(x_r-\mu_{X_{i,\ell}})^2=2(x_k-\mu_{X_{i,\ell}}), \tag{3.30}$$

поскольку $\partial\mu_{X_{i,\ell}}/\partial x_k=1/\ell$ и $\sum_{r=i}^{i+\ell}(x_r-\mu_{X_{i,\ell}})=0$. Поэтому имеем

$$\frac{\partial\sigma_{X_{i,\ell}}}{\partial x_k}=\frac{x_k-\mu_{X_{i,\ell}}}{(\ell-1)\sigma_{X_{i,\ell}}}. \tag{3.31}$$

По определению 3.2 $\hat{x}_j=(x_j-\mu_{X_{i,\ell}})/\sigma_{X_{i,\ell}}$. Дифференцируя это выражение по x_k, получаем:

$$\frac{\partial\hat{x}_j}{\partial x_k}=\frac{1}{\sigma^2_{X_{i,\ell}}}\left(\frac{\partial}{\partial x_k}(x_j-\mu_{X_{i,\ell}})\,\sigma_{X_{i,\ell}}-(x_j-\mu_X)\,\frac{\partial\sigma_{X_{i,\ell}}}{\partial x_k}\right). \tag{3.32}$$

Теперь рассмотрим уменьшаемое второго множителя правой части формулы 3.32:

$$\frac{\partial}{\partial x_k}(x_j-\mu_X)=\frac{\partial x_j}{\partial x_k}-\frac{\partial\mu_{X_{i,\ell}}}{\partial x_k}=\delta_{jk}-\frac{1}{\ell}. \tag{3.33}$$

Тогда, подставив получившееся выражение в формулу 3.32, получаем:

$$\frac{\partial\hat{x}_j}{\partial x_k}=\frac{\delta_{jk}-\frac{1}{\ell}}{\sigma_{X_{i,\ell}}}-\frac{x_j-\mu_{X_{i,\ell}}}{\sigma^2_{X_{i,\ell}}}\cdot\frac{\partial\sigma_{X_{i,\ell}}}{\partial x_k}. \tag{3.34}$$

Подставим формулу 3.31 в выражение 3.32, получаем:

$$\begin{aligned}\frac{\partial\hat{x}_j}{\partial x_k}&=\frac{\delta_{jk}-\frac{1}{\ell}}{\sigma_{X_{i,\ell}}}-\frac{x_j-\mu_{X_{i,\ell}}}{\sigma^2_{X_{i,\ell}}}\cdot\frac{x_k-\mu_{X_{i,\ell}}}{(\ell-1)\sigma_X}=\\&=\frac{\delta_{jk}-\frac{1}{\ell}}{\sigma_{X_{i,\ell}}}-\frac{(x_j-\mu_{X_{i,\ell}})(x_k-\mu_{X_{i,\ell}})}{(\ell-1)\,\sigma^3_{X_{i,\ell}}}.\end{aligned} \tag{3.35}$$

Пусть $\widehat{\mathrm{ED}}(X_{i,\ell},Y_{i,\ell})=\mathrm{ED}(\widehat{X}_{i,\ell},\widehat{Y}_{i,\ell})$. Тогда, поскольку $\partial\hat{y}_{i+j}/\partial x_k=0$, в соответствии с цепным правилом имеем:

$$\frac{\partial}{\partial x_k}\widehat{\mathrm{ED}}(X_{i,\ell},Y_{i,\ell})=\frac{1}{\widehat{\mathrm{ED}}(X_{i,\ell},Y_{i,\ell})}\sum_{j=1}^{\ell}(\hat{x}_{i+j}-\hat{y}_{i+j})\,\frac{\partial\hat{x}_{i+j}}{\partial x_k}. \tag{3.36}$$

Так как MPDE — усреднение вкладов всех окон, частная производная x_k равна сумме вкладов от тех окон, которые содержат k-ю точку:

$$\frac{\partial \mathrm{MPDE}}{\partial x_k} = \frac{1}{2c} \sum_{i=i_{min}}^{i_{max}} \left(\alpha \frac{x_k - y_k}{\mathrm{ED}(X_{i,\ell}, Y_{i,\ell})} + \right.$$
$$\left. + \beta \frac{1}{\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell})} \sum_{j=1}^{\ell} (\hat{x}_{i+j} - \hat{y}_{i+j}) \frac{\partial \hat{x}_{i+j}}{\partial x_k} \right). \tag{3.37}$$
$$i_{\min} = \max(1, k - \ell), i_{\max} = \min(k, m - \ell + 1).$$

Подстановка выражения 3.35 в выражение 3.37 дает искомую формулу. Наконец, если $X = F(Z; \theta)$, тогда в соответствии с цепным правилом имеем:

$$\frac{\partial L}{\partial \theta} = (\nabla_X \mathrm{MPDE}(X, Y))^\top \frac{\partial X}{\partial \theta}, \nabla_X \mathrm{MPDE}(X, Y) = \begin{bmatrix} \frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_1} \\ \frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_2} \\ \vdots \\ \frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_m} \end{bmatrix}. \tag{3.38}$$

□

Утверждение 4. ***Вычислительная сложность.*** *Пусть окна* $X_{i,\ell}, Y_{i,\ell} \in \mathbb{R}^\ell$ *имеют длину* $\ell = \lfloor \gamma m \rfloor$ *для некоторой фиксированной константы* $0 < \gamma < 1$, *а* $c = m - \ell + 1$ *есть число окон. Тогда функция* $\mathrm{MPDE}(X, Y)$ *имеет вычислительную сложность* $O(m^2)$.

Доказательство. Рассмотрим одно окно длины ℓ. Для наивного (прямого) вычисления вклада $\alpha\,\mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) + \beta\,\widehat{\mathrm{ED}}(X_{i,\ell}, Y_{i,\ell})$ выполняются следующие операции:

1. Вычисление средних значений окон по формуле 3.2 требует $O(\ell)$ операций.

2. Вычисление стандартных отклонений окон по формуле 3.2 требует $O(\ell)$ операций.

3. Нормализация точек окон по формуле 3.2 $\hat{X} = (X - \mu_X \mathbf{1})/\sigma_X$ и $\hat{Y}$ требует $O(\ell)$ операций.

4. Вычисление $\mathrm{ED}(X_{i,\ell}, Y_{i,\ell}) = \sqrt{\sum_{j=1}^{\ell}(x_j - y_j)^2 + \varepsilon}$ по формуле 3.1 требует $O(\ell)$ операций.

5. Вычисление $\widehat{\mathrm{ED}}(\hat{X}, \hat{Y})$ также требует $O(\ell)$ операций.

Следовательно, для одного окна длины ℓ количество необходимых вычислений оценивается как $O(\ell)$. Общая сложность вычисления MPDE для всех окон равна

$$O(c \cdot \ell). \tag{3.39}$$

Учитывая, что $\ell = \lfloor \gamma m \rfloor$ и $c = m - \ell + 1 = O(m)$, получаем

$$O(c \cdot \ell) = O(m \cdot m) = O(m^2). \tag{3.40}$$

Таким образом, при указанных предположениях вычисление функции MPDE обладает квадратичной сложностью по m. □

Градиент функции потерь

Градиент функции потерь нейросетевой модели восстановления временных рядов играет ключевую роль при обучении, поскольку определяет направление обновления весов нейросетевой модели. В дальнейшем рассмотрим его более подробно, чтобы оценить влияние различных компонентов функции потерь на процесс обучения. Частная производная MPDE по переменной x_i вычисляется как среднее значение производных расстояний между окнами, содержащими точку x_i:

$$\frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_i} = \frac{1}{2 \cdot c} \sum_{j=j_{\min}}^{j_{max}} \Big(\frac{\partial \mathrm{ED}(X_{j,\ell}, Y_{j,\ell})}{\partial x_i} + \frac{\partial \widehat{\mathrm{ED}}(X_{j,\ell}, Y_{j,\ell})}{\partial x_i} \Big),$$
$$j_{\min} = \max(1, i - \ell), j_{\max} = \min(i, m - \ell + 1). \tag{3.41}$$

Учитывая формулы 3.1 и 3.2, частная производная функции потерь по переменной x_i может быть выражена следующим образом:

$$\begin{gathered}\frac{\partial \mathrm{MPDE}(X,Y)}{\partial x_i} = A + B + C,\\ A = \frac{\alpha}{2\cdot c}\cdot\sum_{j_{\min}}^{j_{\max}}\frac{x_i - y_i}{\mathrm{ED}(X_{j,\ell},Y_{j,\ell})},\ B = \frac{\beta}{2\cdot c}\cdot\sum_{j_{\min}}^{j_{\max}}\frac{\widehat{x}_i - \widehat{y}_i}{\widehat{\mathrm{ED}}(X_{j,\ell},Y_{j,\ell})},\\ C = \frac{\beta}{2\cdot\ell\cdot c}\cdot\sum_{j_{\min}}^{j_{\max}}\frac{\partial\widehat{\mathrm{ED}}(X_{j,\ell},Y_{j,\ell})}{\partial\mu_j}\cdot\frac{\partial\mu_j}{\partial x_i} +\\ +\frac{\beta}{2\cdot\sigma\cdot c}\cdot\sum_{j_{\min}}^{j_{\max}}\frac{\partial\widehat{\mathrm{ED}}(X_{j,\ell},Y_{j,\ell})}{\partial\sigma_j}\cdot\frac{\partial\sigma_j}{\partial x_i}.\end{gathered} \tag{3.42}$$

Рассмотрим слагаемые A, B и C в данной формуле более подробно. Слагаемое A вычисляется как сумма разностей между точками восстановленного и истинного временного ряда, деленных на евклидово расстояние соответствующих окон. Данное слагаемое отвечает за минимизацию отклонений между восстановленными и истинными значениями, обеспечивая корректное воспроизведение масштаба и среднего значения исходного временного ряда. Деление на евклидово расстояние масштабирует вклад каждой точки в градиент пропорционально размеру ошибки, предотвращая чрезмерно большие обновления весов сети при большом значении ошибки. Разбиение подпоследовательности на окна позволяет масштабировать градиент на уровне локальных фрагментов. Поскольку градиенты вычисляются отдельно для каждого окна и масштабируются в зависимости от локальной ошибки, влияние выбросов или аномальных значений локализуется и не распространяется на другие части последовательности. В результате это предотвращает искажение обновлений весов нейросетевой модели для остальных окон подпоследовательности.

Для воспроизведения поведенческих особенностей подпоследовательностей в слагаемом B используются z-нормализованные окна. Нормализация устраняет влияние масштаба и среднего значения, и делает сопоставимыми подпоследовательности с различной амплитудой и смещением. В результа-

те сравниваются относительные изменения точек внутри окна, включая локальные тренды, пики и впадины. В отличие от работы с ненормализованными данными, где минимизация ошибки может происходить за счет изменения масштаба и смещения, слагаемое B стимулирует нейросетевую модель воспроизводить поведенческие особенности подпоследовательностей.

Слагаемое C отражает косвенное влияние каждой точки на нормализацию всего окна подпоследовательности. Поскольку параметры нормализации (μ и σ) зависят от всех точек окна, изменение одного предсказанного значения влияет на нормализованные значения остальных. Отклонение восстановленного значения от истинного смещает статистические параметры окна и, как следствие, нормализованные ошибки других точек изменяются, создавая каскадный эффект. Слагаемое C учитывает производные параметров нормализации по каждой точке. В результате функция потерь учитывает влияние отдельного предсказания на весь контекст окна и стимулирует нейросетевую модель корректно учитывать, как ошибка в одной точке отражается на нормализацию и восстановлении значений остальных точек подпоследовательности.

Изменение весовых коэффициентов α и β позволяет гибко регулировать влияние соответствующих компонентов функции потерь на процесс обучения модели. Коэффициент α контролирует вклад слагаемого A, отвечающего за сравнение отдельно взятых точек. Увеличение α делает нейросетевую модель более чувствительной к ошибкам между отдельно взятыми точками.

Коэффициент β контролирует совместный вклад слагаемых B и C, которые оценивают поведенческое сходство подпоследовательностей. Повышение β усиливает внимание модели к сохранению поведенческих особенностей ряда и улучшает внутреннюю согласованность точек подпоследовательности. Настройка α и β позволяет балансировать между точностью восстановления отдельных значений и качеством воспроизведения формы и структуры временного ряда.

Случай многомерного ряда

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

$$\mathrm{MPDE}(\boldsymbol{X},\ \boldsymbol{Y}) = \frac{1}{d}\sum_{i=1}^{d}\mathrm{MPDE}(X^{(i)}, Y^{(i)}). \tag{3.43}$$

3.1.2. Параллельный алгоритм вычисления

В процессе обучения нейросетевых моделей временных рядов вычисление функции потерь производится на совокупности нескольких подпоследовательностей, объединенных в батч (batch). Применение батчевой обработки позволяет одновременно обрабатывать несколько примеров, повышая вычислительную эффективность через параллельную обработку подпоследовательностей. Другим преимуществом использования батчевой обработки является усреднение ошибок по набору элементов обучающей выборки. Усреднение ошибок стабилизирует процесс обучения, так как обновления параметров модели становятся более равномерными и менее чувствительными к выбросам. Для обеспечения практической применимости функции потерь MPDE в процессе обучения нейросетевых моделей восстановления временных рядов в данной работе предлагается параллельный алгоритм ее вычисления. В качестве входных данных MPDE будем полагать два батча (восстановленных $\boldsymbol{X}$ и истинных $\boldsymbol{Y}$ подпоследовательностей), состоящих из b многомерных подпоследовательностей длиной m и содержащих d измерений. Результатом работы полагается значение функции потерь, усредненное по всем элементам батча.

Для вычисления MPDE из всех выходных подпоследовательностей необходимо извлечь все окна длины ℓ. Для каждого батча может быть сформирован такой тензор $\boldsymbol{X}_{\text{slice}}$, который содержит множество всех окон для каждой подпоследовательности в батче. Введем функцию unfold_{ℓ}, которая каждому батчу $\boldsymbol{X} \in \mathbb{R}^{b\times m\times d}$ сопоставляет четырехмерный тензор $\boldsymbol{X}_{\text{slice}} \in$

$\mathbb{R}^{b\times(m-\ell-1)\times\ell\times d}$:

$$\begin{gathered}\text{unfold}_\ell : \mathbb{R}^{b\times m\times d} \to \mathbb{R}^{b\times(m-\ell-1)\times\ell\times d}, \quad \boldsymbol{X}_{\text{slice}} = \text{unfold}_\ell(\boldsymbol{X}),\\ \boldsymbol{X}_{\text{slice}}(i,j,p,k) = \boldsymbol{X}(i,j+p,k), \quad 1 \leqslant p \leqslant \ell,\\ 1 \leqslant i \leqslant b, \quad 1 \leqslant j \leqslant m-\ell-1, \quad 1 \leqslant \ell < m, \quad 1 \leqslant k \leqslant d.\end{gathered} \tag{3.44}$$

Для вычисления MPDE из $\boldsymbol{X}$ и $\boldsymbol{Y}$ с помощью функции unfold_ℓ формируются тензоры $\boldsymbol{X}_{slice}$ и $\boldsymbol{Y}_{slice}$, содержащие все окна подпоследовательностей. Перед вычислением нормализованного евклидова расстояния все извлеченные окна предварительно нормализуются по каждому измерению. Для этого вводятся вспомогательные функции, вычисляющие статистические показатели вдоль временной оси массива окон:

$$\begin{gathered}\text{sum}_{\text{axis}} : \mathbb{R}^{b\times(m-\ell-1)\times\ell\times d} \to \mathbb{R}^{b\times(m-\ell-1)\times d}, \quad \boldsymbol{X}_{\text{sum}} = \text{sum}_{\text{axis}}(\boldsymbol{X}_{\text{slice}}),\\ \boldsymbol{X}_{\text{sum}}(i,j,k) = \sum_{p=1}^{\ell} \boldsymbol{X}(i,j,p,k),\end{gathered} \tag{3.45}$$

$$\begin{gathered}\text{mean}_{\text{axis}} : \mathbb{R}^{b\times(m-\ell-1)\times\ell\times d} \to \mathbb{R}^{b\times(m-\ell-1)\times d},\\ \boldsymbol{X}_{\text{mean}} = \text{mean}_{\text{axis}}(\boldsymbol{X}_{\text{slice}}),\ \boldsymbol{X}_{\text{mean}}(i,j,k) = \frac{\boldsymbol{X}_{\text{sum}}(i,j,k)}{\ell},\end{gathered} \tag{3.46}$$

$$\begin{gathered}\text{var}_{\text{axis}} : \mathbb{R}^{b\times(m-\ell-1)\times\ell\times d} \to \mathbb{R}^{b\times(m-\ell-1)\times d}, \quad \boldsymbol{X}_{\text{var}} = \text{var}_{\text{axis}}(\boldsymbol{X}_{\text{slice}}),\\ \boldsymbol{X}_{\text{var}}(i,j,k) = \frac{1}{\ell}\sum_{p=1}^{\ell}(\boldsymbol{X}(i,j,p,k) - \boldsymbol{X}_{\text{mean}}(i,j,k))^2,\\ 1 \leqslant i \leqslant b, \quad 1 \leqslant j \leqslant m-\ell-1, \quad 1 \leqslant \ell < m, \quad 1 \leqslant k \leqslant d,\end{gathered} \tag{3.47}$$

где $\text{mean}_{\text{axis}}$ — функция, вычисляющая среднее значение вдоль оси, var_{axis} — функция, вычисляющая дисперсию вдоль оси и sum_{axis} — возвращающая сумму значений по каждому окну.

На основе формул 3.46 и 3.47 определим тензор $\boldsymbol{X}_{\text{norm}} \in \mathbb{R}^{b\times(m-\ell-1)\times\ell\times d}$, содержащий нормализованные окна всех подпоследовательностей батча:

$$\begin{gathered}\boldsymbol{X}_{\text{norm}}(i,j,p,k) = \frac{\boldsymbol{X}_{\text{slice}}(i,j,p,k) - \boldsymbol{X}_{\text{mean}}(i,j,k)}{\sqrt{\boldsymbol{X}_{\text{var}}(i,j,k) + \varepsilon}}, \\ 1 \leqslant i \leqslant b,\ 1 \leqslant j \leqslant m-\ell+1,\ 1 \leqslant p \leqslant \ell,\ 1 \leqslant k \leqslant d,\end{gathered} \tag{3.48}$$

где $\varepsilon > 0$ малое положительное число, добавляемое для предотвращения деления на ноль (см. формулу 3.2). Аналогично $\boldsymbol{X}_{\text{norm}}$ определим $\boldsymbol{Y}_{\text{norm}}$:

$$\begin{gathered}\boldsymbol{Y}_{\text{norm}}(i,j,p,k) = \frac{\boldsymbol{Y}_{\text{slice}}(i,j,p,k) - \boldsymbol{Y}_{\text{mean}}(i,j,k)}{\sqrt{\boldsymbol{Y}_{\text{var}}(i,j,k) + \varepsilon}}, \\ 1 \leqslant i \leqslant b,\ 1 \leqslant j \leqslant m-\ell+1,\ 1 \leqslant p \leqslant \ell,\ 1 \leqslant k \leqslant d,\end{gathered} \tag{3.49}$$

где $\boldsymbol{Y}_{\text{mean}} = \text{mean}_{\text{axis}}(\boldsymbol{Y}_{\text{slice}})$, $\boldsymbol{Y}_{\text{var}} = \text{var}_{\text{axis}}(\boldsymbol{Y}_{\text{slice}})$ определяются аналогично соответствующим величинам для $\boldsymbol{X}$.

После формирования тензоров, содержащих нормализованные окна, вычисляются тензор евклидовых расстояний $\text{ED} \in \mathbb{R}^{b\times(m-\ell-1)\times d}$ и тензор нормализованных евклидовых расстояний $\text{ED}_{\text{norm}} \in \mathbb{R}^{b\times(m-\ell-1)\times d}$ следующим образом:

$$\begin{gathered}\text{ED}(i,j,k) = \sqrt{\Big(\sum_{p=1}^{\ell} \boldsymbol{Y}_{\text{slice}}(i,j,p,k) - \boldsymbol{X}_{\text{slice}}(i,j,p,k)\Big)^2 + \varepsilon}, \\ \text{ED}_{\text{norm}}(i,j,k) = \sqrt{\Big(\sum_{p=1}^{\ell} \boldsymbol{Y}_{\text{norm}}(i,j,p,k) - \boldsymbol{X}_{\text{norm}}(i,j,p,k)\Big)^2 + \varepsilon}, \\ 1 \leqslant i \leqslant b,\ 1 \leqslant j \leqslant m-\ell+1,\ 1 \leqslant k \leqslant d,\ \varepsilon > 0.\end{gathered} \tag{3.50}$$

На последнем шаге вычисляется значение функции потерь $\text{MPDE} \in \mathbb{R}$ как среднее значение по батчу взвешенной суммы евклидова и нормализованного евклидова расстояний между соответствующими окнами восста-

новленных и истинных окон подпоследовательностей:

$$\begin{gathered}\mathrm{MPDE} = \frac{1}{2p}\sum_{i=1}^{b}\sum_{j=1}^{m-\ell-1}\sum_{k=1}^{d}(\alpha\mathrm{ED} + \beta\mathrm{ED}_{\mathrm{norm}}),\\ p = b\cdot d\cdot(m-\ell-1).\end{gathered} \tag{3.51}$$

Алг. 3.1. TorchMPDE(IN: $\boldsymbol{X}$, $\boldsymbol{Y} \in \mathbb{R}^{b\times m\times d}$, α, β; OUT: MPDE)

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

Реализация вычисления функции потерь MPDE представлена в алг. 3.1. Для параллельной реализации вычисления функции потерь, описанной выше, используется фреймворк PyTorch [105], который в настоящее время является одним из стандартных средств реализации нейросетевых моделей. Основной структурой для хранения данных в PyTorch является тензор (Tensor), представляющий собой многомерный тензор вещественных значений. Фреймворк обеспечивает выполнение базовых операций над тензорами (вычисление среднего, дисперсии, стандартного отклонения и др.) с поддержкой автоматического дифференцирования и параллельного выполнения на графических процессорах [23].

Для реализации используются базовые методы, выполняющие два вида операций: агрегация и извлечение данных. Методы агрегации осуществляют вычисление среднего (`mean`), суммы (`sum`), квадратного корня (`sqrt`) и дисперсии (`var`) над входным тензором. Операции могут выполняться по всем измерениям тензора или вдоль выбранного измерения. Соответствие между используемыми выше математическими функциями и методами PyTorch следующее: среднее значение по строкам тензора $\boldsymbol{X}$, обозначаемое как $\mathrm{mean}_{\mathrm{axis}}(\boldsymbol{X})$, соответствует вызову $\texttt{mean}(\boldsymbol{X},\, \mathrm{dim}{=}2)$ в PyTorch;

сумма по строкам $\mathrm{sum}_{\mathrm{axis}}(\boldsymbol{X})$ реализуется как $\texttt{sum}(\boldsymbol{X}, \mathrm{dim}{=}2)$; дисперсия по строкам $\texttt{var}(\boldsymbol{X})$ соответствует $\mathrm{var}(\boldsymbol{X}, \mathrm{dim}{=}2)$; вычисление квадратного корня осуществляется с помощью функции $\texttt{sqrt}(\boldsymbol{X})$. При этом аргумент `dim` указывает ось, вдоль которой производится агрегирование, что соответствует выбору оси в математических формулах.

Для извлечения данных используется метод `unfold`, который позволяет выделять скользящие окна вдоль заданной размерности тензора. В контексте реализации MPDE данный метод применяется для формирования всех окон длины ℓ, что соответствует действию функции unfold_{ℓ}.

3.2. Метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления

В данном разделе представлен новый метод tsGAP (time-series Graph-Attention Perfomance Prediction model), предназначенный для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Предлагаемый метод может применяться как инструмент предварительной оценки на этапе выбора целевой модели. Входными данными метода являются архитектура целевой нейросетевой модели и вектор параметров ее обучения. Выходными данными метода являются вектора из двух параметров: ошибка целевой нейросетевой модели и время ее обучения. Метод реализован с помощью нейронной сети, состоящей из нескольких компонентов: Автоэнкодер, кодирующий архитектуру целевой модели в векторное представление, Энкодер параметров обучения и Агрегатор, формирующий итоговый прогноз.

3.2.1. Нейросетевой метод tsGAP

В рамках данной главы анализируемую нейросетевую модель будем называть *целевой*. *Целевая нейросетевая модель* может быть формально представлена как ориентированный ациклический граф, в котором верши-

ны соответствуют слоям модели, а дуги представляют собой направленные связи между слоями. Введем набор понятий, необходимых для формального представления целевой нейросетевой модели.

Набор типов слоев $\mathcal{C}$ представляет собой упорядоченный набор, состоящий из c элементов. Каждый элемент набора представляет собой целочисленный код, соответствующий допустимому типу нейросетевого слоя или операции, используемой в модели:

$$\mathcal{C} = \{C_i\}_{i=1}^{c}, \quad C_i \in \mathbb{Z}. \tag{3.52}$$

Для каждого допустимого элемента $\mathcal{C}$ может быть задано до двух числовых параметров, определяющих его конфигурацию и функциональные характеристики. Перечень типов слоев, операций и их параметров приведен в табл. С.3.

Набор функций активации $\mathcal{A}$ представляет собой упорядоченный набор из a элементов. Каждый элемент набора соответствует допустимой функции активации, применяемой в слоях нейронной модели:

$$\mathcal{A} = \{A_i\}_{i=1}^{a}, \quad A_i \in \mathbb{Z}. \tag{3.53}$$

Перечень используемых функций активации приведен в табл. С.1.

Формально целевая нейросетевая модель может быть представлена парой объектов: упорядоченный набор слоев L и матрица связей $R \in \mathbb{R}^{\lambda \times 2}$. Рассмотрим каждый элемент более подробно. Набор слоев L содержит ℓ четырехэлементных кортежей, каждый из которых описывает один слой нейронной сети. Формально каждый слой-кортеж может быть представлен следующим образом:

$$\begin{gathered} L_k = (C_i, p_1, p_2, A_j),\ 1 \leqslant i \leqslant c, \\ 1 \leqslant j \leqslant a,\ p_1, p_2 \in \mathbb{R},\ 1 \leqslant k \leqslant \ell,\ C_i \in \mathcal{C},\ A_j \in \mathcal{A}, \end{gathered} \tag{3.54}$$

где C_i — целочисленный код типа слоя, p_1 и p_2 — числовые параметры слоя, A_j — целочисленный код функции активации.

Связи между слоями нейронной модели задаются матрицей R, каждая строка которой соответствует одной дуге ориентированного графа. Каждая дуга указывает на направление передачи данных от одного слоя к другому. Каждая связь описывается парой индексов: индекс исходного слоя и индекс целевого слоя. Формально каждая строка матрицы связей может быть определена следующим образом:

$$R(i,\cdot) = (r_j, r_k), \quad 1 \leqslant i \leqslant \lambda, \quad 1 \leqslant r_j, r_k \leqslant \ell, \quad i < j, \tag{3.55}$$

где r_j — индекс слоя-источника, r_k — индекс целевого слоя, λ — общие количество связей в нейросетевой модели.

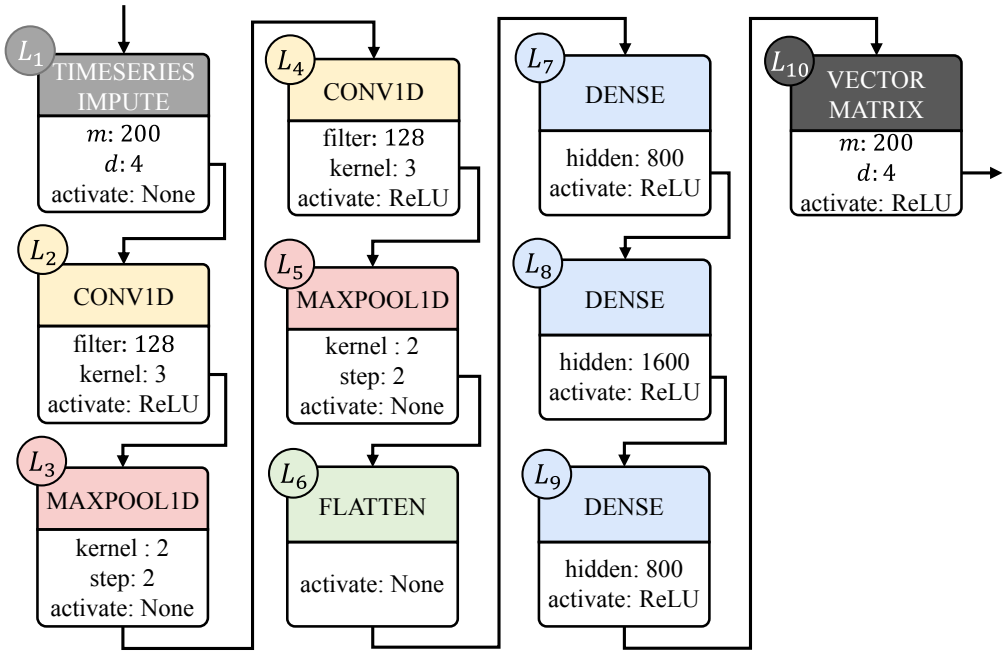


Рис. 3.1. Пример графового представления целевой нейросетевой модели

Для наглядности рассмотрим пример кодирования целевой нейросетевой модели. На рис. 3.1 нейросетевая модель изображена в виде ориентированного ациклического графа. Каждый узел графа соответствует одному слою (сверточному, полносвязному или др.). Направление дуги отражает порядок вычислений и поток данных между ними.

Таблица 3.1 демонстрирует набор слоев L, построенный на основе данного графа. Каждая строка таблицы описывает один слой нейросети и со-

Табл. 3.1. Пример набора слоев целевой модели

Слой	Название	Первый параметр	Второй параметр	Функция активации
L_1	TIMESERIESIMPUTE	200	4	NONE
L_2	CONV1D	128	3	RELU
L_3	MAXPOOL1D	2	2	NONE
L_4	CONV1D	64	3	RELU
L_5	MAXPOOL1D	2	2	NONE
L_6	FLATTEN	0	0	NONE
L_7	DENSE	800	0	RELU
L_8	DENSE	1600	0	RELU
L_9	DENSE	800	0	RELU
L_{10}	VECTORMATRIX	200	4	NONE

держит сведения о его типе, параметрах и используемой функции активации.

Введем операцию среза slice, которая заключается в выделении из исходной матрицы подматрицы, ограниченной заданными диапазонами индексов строк и столбцов:

$$\begin{gathered}
\mathrm{slice}_{i:j,k:v} : \mathbb{R}^{b\times q} \to \mathbb{R}^{(j-i+1)\times(v-k+1)}, \\
\mathrm{slice}_{i:j,k:v}(A) = \begin{bmatrix} A_{i,k} & A_{i,k+1} & \cdots & A_{i,v} \\ A_{i+1,k} & A_{i+1,k+1} & \cdots & A_{i+1,v} \\ \vdots & \vdots & \ddots & \vdots \\ A_{j,k} & A_{j,k+1} & \cdots & A_{j,v} \end{bmatrix}, \\
1 \leqslant i \leqslant j \leqslant b, \quad 1 \leqslant k \leqslant v \leqslant q,
\end{gathered} \tag{3.56}$$

где $A \in \mathbb{R}^{b\times q}$ — исходная матрица, $[i, j]$ и $[k, v]$ обозначают диапазоны индексов строк и столбцов соответственно. Результатом операции является матрица, содержащая элементы исходной матрицы с номерами строк от i до j и столбцов от k до v, включительно.

Частным случаем операции среза является срез по столбцам $\mathrm{slice}_{:,k:v}$. Результатом такой операции является подматрица, содержащая те же строки, что и исходная матрица, и только те столбцы, номера которых при-

надлежат интервалу $[k, v]$. Другим частным случаем является одномерный срез строки $\text{slice}_{i,k:v}$, возвращающий вектор, содержащий элементы строки i, расположенные в столбцах с номерами от k до v, включительно.

3.2.2. Компоненты метода

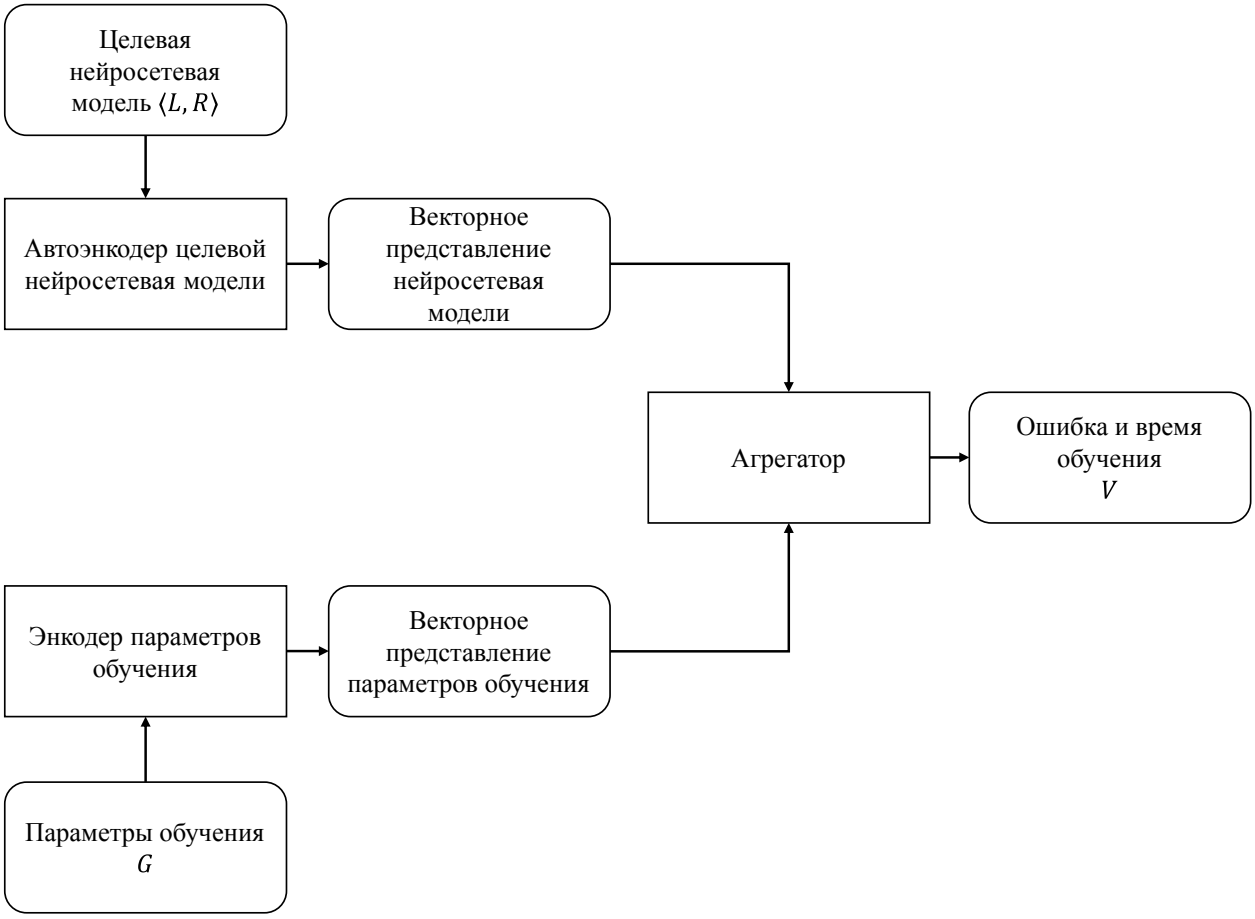


Рис. 3.2. Метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления

Метод, представленный на рис. 3.2, включает следующие компоненты, последовательно обрабатывающие входные данные: Автоэнкодер целевой нейросетевой модели, Энкодер параметров обучения и Агрегатор признаков. На вход Автоэнкодеру поступает графовое представление целевой нейросетевой модели, состоящее из двух элементов: набора слоев L и матрицы связей R. В процессе обработки входных данных Автоэнкодер формирует векторное представление целевой нейросетевой модели, обозначаемое как

$Z \in \mathbb{R}^z$. Энкодер принимает на вход параметры обучения целевой нейросетевой модели и формирует их векторное представление. Полученные векторные представления целевой нейросетевой модели и параметров обучения передаются на вход Агрегатору, который на выходе продуцирует прогноз в виде вектора из двух значений: минимальной ошибки нейросетевой модели на валидации и времени выполнения одной эпохи обучения.

Предварительная обработка целевой нейросетевой модели

Перед подачей на вход Автоэнкодера каждый слой из набора слоев L проходит предварительную обработку, включающую следующие этапы: заполнение, нормализация параметров, кодирование типов слоев и кодирование функций активаций. В результате предварительной обработки набор слоев L преобразуется в матрицу слоев $\widehat{L} \in \mathbb{R}^{\ell \times (c+a+2)}$. На этапе заполнения входной набор слоев L приводится к фиксированной длине ℓ путем добавления специальных фантомных слоев с типом `NONE`, обозначающих отсутствие реального слоя. Новые слои не содержат ни параметров, ни функций активации.

Числовые параметры слоя p_1 и p_2 перед подачей на вход нейронной сети подвергаются минимаксной нормализации:

$$\widehat{p} = \frac{p - p_{\min}}{p_{\max} - p_{\min}}, \tag{3.57}$$

где p — параметр слоя, $p_{\min}$ и $p_{\max}$ — минимальное и максимальное значения данного параметра среди всех слоев одного и того же типа.

Целочисленные коды типа слоя C_i и функции активации A_j преобразуются с использованием one-hot кодирования. Введем функцию onehot_n, которая отображает целое число $k \in \{0, \ldots, n-1\}$ в бинарный вектор длины n, где единственная единица расположена на k-й позиции:

$$\text{onehot}_n(k) : \mathbb{Z} \to \{0,1\}^n, \quad \text{onehot}_n(k)_i = \begin{cases} 1, i = k, \\ 0, \text{иначе.} \end{cases} \tag{3.58}$$

Нормализованные параметры, one-hot коды типов слоев и функций активации конкатенируются в вектора. Сцепление таких векторов формирует нормализованную матрицу слоев $\widehat{L}$. Каждая строка нормализованной матрицы $\widehat{L}$ соответствует одному слою и может быть формально представлена следующим образом:

$$\begin{gathered}\widehat{L}(k,\cdot) = \mathrm{onehot}_c(C_i) \cdot (\widehat{p}_1, \widehat{p}_2) \cdot \mathrm{onehot}_a(A_j), \\ 1 \leqslant k \leqslant \ell, \quad 1 \leqslant i \leqslant c, \quad 1 \leqslant j \leqslant a,\end{gathered} \tag{3.59}$$

где символ «$\cdot$» обозначает операцию конкатенации.

Аналогично матрице слоев, матрица связей R преобразуется в нормализованную матрицу связей. Для получения нормализованной матрицы связей $\widehat{R} \in \mathbb{R}^{\lambda \times 2\ell}$ применяется one-hot кодирование индексов:

$$\begin{gathered}\widehat{R}(k,\cdot) = \mathrm{onehot}_\ell(R(k,i)) \cdot \mathrm{onehot}_\ell(R(k,j)), \\ 1 \leqslant k \leqslant \lambda, \quad 1 \leqslant i,j \leqslant \ell.\end{gathered} \tag{3.60}$$

Формирование векторного представления целевой нейросетевой модели

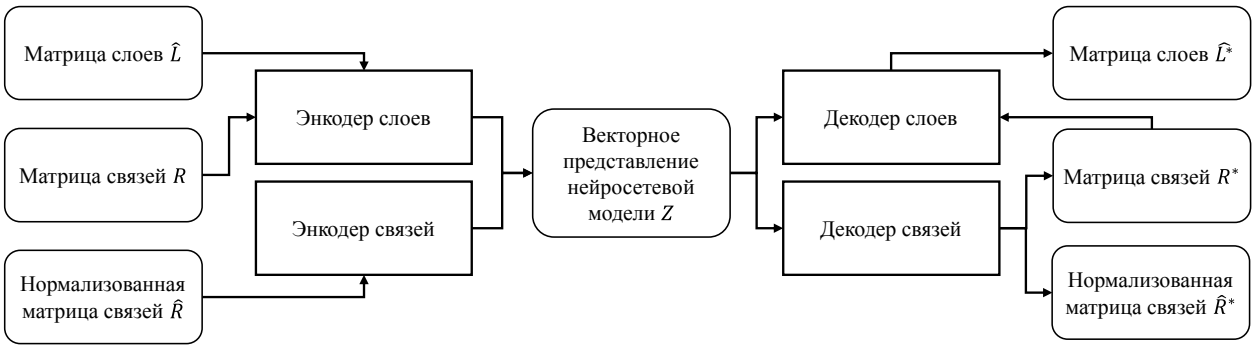


Рис. 3.3. Архитектура нейросетевой модели для формирования векторного представления целевой модели

Формирование векторного представления целевой нейросетевой модели осуществляется с помощью Автоэнкодера. Нормализованная матрица слоев $\widehat{L}$, исходная матрица связей R и ее нормализованная версия $\widehat{R}$ по-

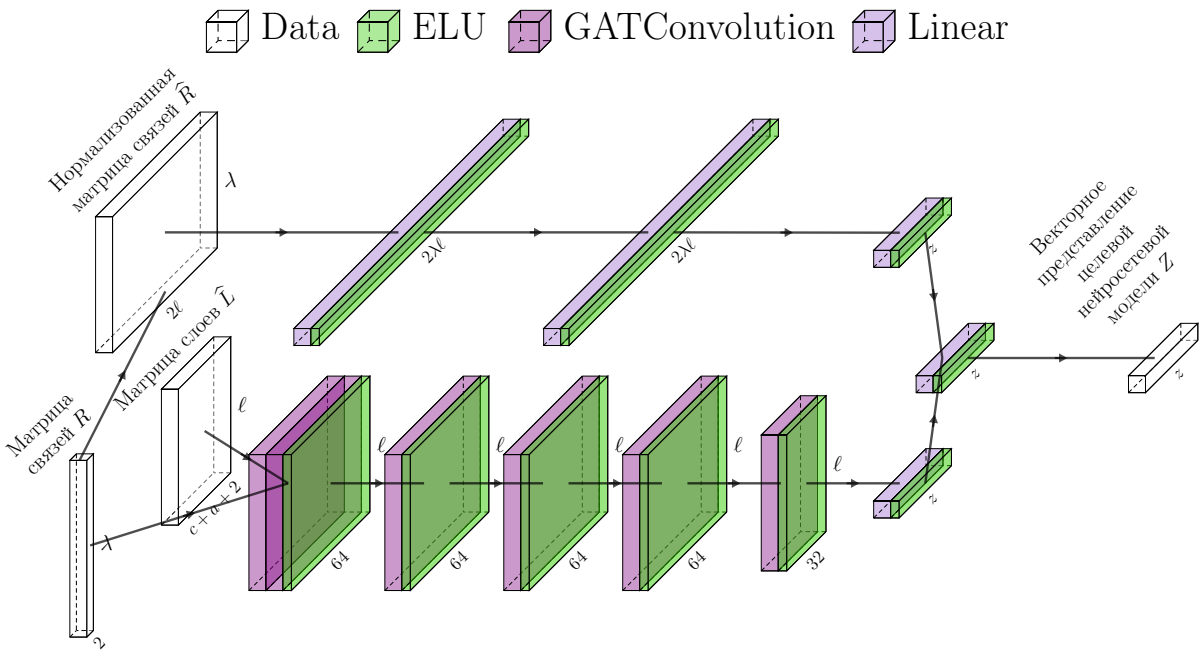


Рис. 3.4. Энкодеры нейросетевой модели для формирования векторного представления целевой модели

даются на вход Автоэнкодера, структура которого показана на рис. 3.3. Автоэнкодер включает четыре подсети.

Первые две подсети (см. рис. 3.4) представляют собой Энкодер слоев и Энкодер связей, которые отвечают за построение векторного представления целевой нейросетевой модели $Z \in \mathbb{R}^z$. Оставшиеся две подсети (см. рис. 3.5) представляют собой Декодер слоев и Декодер связей. Декодеры производят восстановление графового представления нейросетевой модели из ее векторного представления. В результате работы декодеров формируются декодированные версии матрицы слоев $\widehat{L}^*$, матрицы связей R^* и ее нормализованной версии $\widehat{R}^*$. Далее рассматривается каждая подсеть подробнее.

Энкодер связей принимает на вход нормализованную матрицу связей $\widehat{R}$ и преобразует ее в векторное представление размерности z. Процесс формирования выходного вектора реализован посредством последовательного прохождения данных через три полносвязных слоя. Первые два слоя содержат по $2\lambda\ell$ нейрона каждый. Последний слой, состоящий из z нейронов,

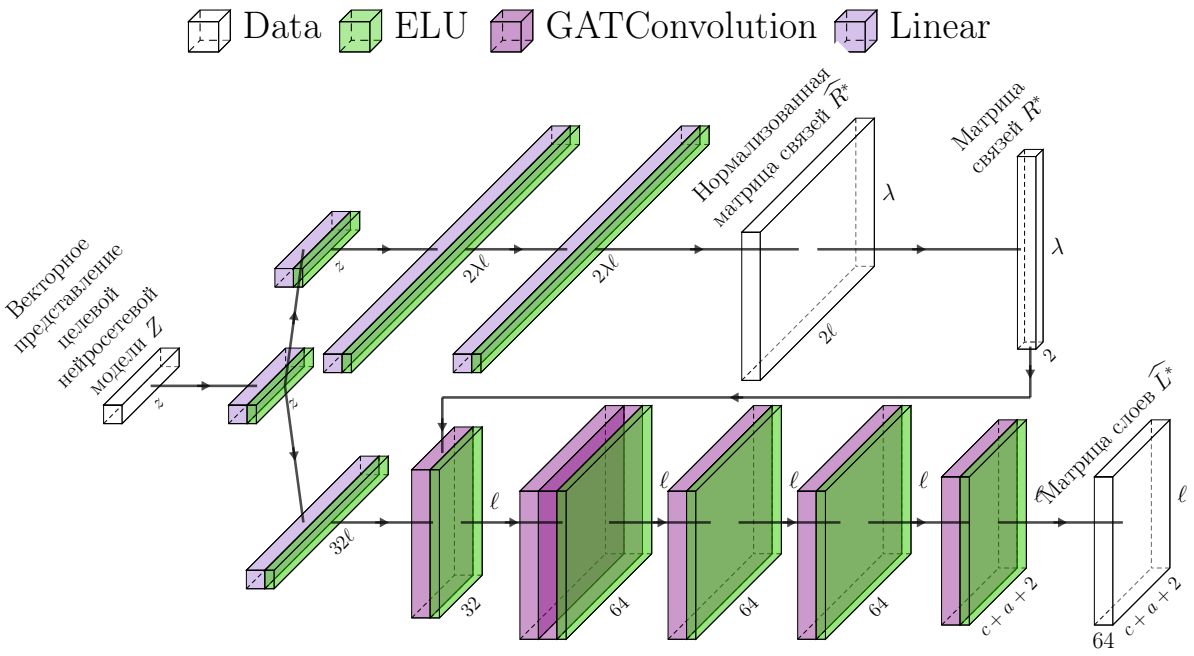


Рис. 3.5. Декодеры нейросетевой модели для формирования векторного представления целевой модели

формирует итоговое векторное представление связей целевой нейросетевой модели.

Энкодер слоев принимает на вход нормализованную матрицу слоев и матрицу связей. На выходе данной подсети формируется векторное представление слоев. Поскольку целевая нейросетевая модель представлена в виде ориентированного ациклического графа, для ее обработки используются графовые нейронные сети. В предлагаемом методе применяется модификация графовой сверточной сети с механизмом внимания (Graph Attention Convolution, GATConv) [133]. Для получения векторного представления применяется последовательность из пяти GATConv слоев и одного полносвязного слоя. Первые четыре слоя GATConv обладают размерностью скрытого представления, равной 64, тогда как последний слой имеет размерность 32. Полносвязный слой, состоящий из z нейронов, принимает выход последнего GATConv слоя и формирует векторное представление слоев целевой модели.

Векторные представления слоев и связей целевой нейросетевой модели объединяются с помощью операции конкатенации. Полученный вектор подается на вход полносвязного слоя, содержащего z нейронов. Данный слой формирует итоговое векторное представление модели, обозначаемое как Z. Во всех упомянутых слоях в качестве функции активации используется Exponential Linear Unit (ELU) [37].

Для декодирования целевой нейросетевой модели векторное представление Z поступает на вход одному полносвязному слою, содержащему z нейронов, который формирует декодированную версию данного представления. Выход данного слоя поступает на вход Декодеру слоев и Декодеру связей.

Декодер связей с помощью трех последовательно применяемых полносвязных слоев формирует декодированную матрицу связей $R^* \in \mathbb{R}^{\lambda \times 2}$. Каждый из этих слоев содержит $2\lambda\ell$ нейронов. В качестве функции активации для первых двух слоев используются Exponential Linear Unit (ELU), тогда как функцией активации последнего слоя является softmax. На выходе подсети формируется декодированная матрица $\widehat{R}^*$, в которой каждая строка представляет собой конкатенацию двух векторов длины ℓ. Первый вектор содержит вероятности того, что соответствующий слой выступает источником данных, а второй вероятности участия каждого слоя в качестве целевого. Для получения итоговой декодированной матрицы связей R^* к каждому такому вектору применяется операция $\arg\max$. Данная операция преобразует вероятностные оценки в индексы, выбирая для каждой связи наиболее вероятные начальный и целевой слои.

На вход Декодера слоев подается декодированная версия векторного представления и декодированная матрица связей R^*. На первом этапе декодирования векторное представление обрабатывается полносвязным слоем, содержащим 32ℓ нейронов. Далее, с помощью пяти последовательно применяемых графовых слоев GATConv, на основе выхода предыдущего слоя и декодированной матрицы связей формируется декодированная матрица слоев $\widehat{L}^*$. Первые четыре слоя имеют размер скрытого представления, равный 64. Последний слой формирует окончательное представление сло-

ев и имеет размер скрытого представления, равный $c + a + 2$. Во втором GATConv используется две головы внимания, в остальных используется по одной. В качестве функции активации для всех слоев, кроме последнего, применяется ELU.

Функция активации последнего слоя является составной. Пусть выход последнего слоя Декодера слоев имеет вид матрицы $O \in \mathbb{R}^{\ell \times (c+a+2)}$, где каждая строка соответствует одному слою целевой нейросетевой модели, а столбцы представляют собой различные декодированные характеристики слоя. К первым c столбцам каждой строки применяется функция softmax для получения распределения вероятностей принадлежности слоя к различным типам из набора $\mathcal{C}$. Следующие два столбца каждой строки содержат числовые параметры слоя, к которым не применяется функция активации. Последние a столбцов каждой строки содержат значения, которые после применения функции softmax преобразуются в значения, отражающие вероятности наличия у данного слоя функции активации из набора $\mathcal{A}$. Формально строка декодированной матрицы слоев $\widehat{L}^*$ может быть представлена следующим образом:

$$\begin{gathered}\widehat{L}^*(i,\cdot) = \operatorname{softmax}(\operatorname{slice}_{i,1:c}(O)) \cdot \operatorname{slice}_{i,c:w}(O) \cdot \operatorname{softmax}(\operatorname{slice}_{i,w+1:s}(O)), \\ 1 \leqslant i \leqslant \ell, \quad w = c+2, \quad s = c+a+2,\end{gathered} \tag{3.61}$$

где функция $\operatorname{softmax} : \mathbb{R}^m \to \mathbb{R}^m$ для вектора $X = (x_1, \ldots, x_m)$ определяется следующим образом:

$$\operatorname{softmax}(X)_j = \frac{e^{x_j}}{\sum_{k=1}^{m} e^{x_k}}, \quad 1 \leqslant j \leqslant m. \tag{3.62}$$

Формирование векторного представления параметров обучения

На рис. 3.6 представлена архитектура Энкодера параметров обучения. На вход Энкодеру поступает вектор $G \in \mathbb{R}^{10}$, содержащий совокупность числовых признаков двух категорий: гиперпараметры обучения и характеристики вычислительного устройства, на котором осуществляется обу-

чение целевой нейросетевой модели. Полный список подаваемых на вход параметров приведен в табл. С.2.

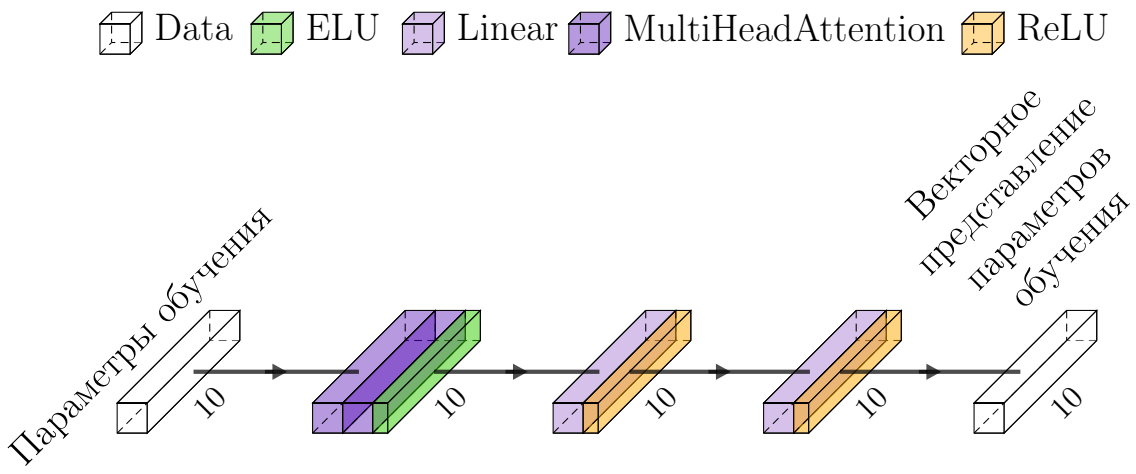


Рис. 3.6. Архитектура модели для формирования векторного представления параметров обучения

Архитектура Энкодера параметров обучения включает следующие последовательно применяемые слои: слой с многоголовым механизмом внимания (multi-head attention) и два полносвязных слоя. Механизм внимания используется для выявления взаимосвязей между компонентами вектора G и определения их значимости при формировании итогового представления. Размер скрытого представления в слое с вниманием составляет 10, количество голов равно 2. Полносвязные слои содержат по 10 нейронов каждый и формируют векторное представление параметров обучения. В качестве функции активации во всех полносвязных слоях используется Rectified Linear Unit (ReLU) [61].

Формирование прогноза

На рис. 3.7 представлена архитектура нейросетевой модели, реализующей компонент Агрегатор. В качестве входных данных Агрегатор получает векторные представления целевой нейросетевой модели и параметров ее обучения. На выходе Агрегатора формируются вектор $V \in \mathbb{R}^2$, который содержит прогнозируемые значения характеристик качества модели.

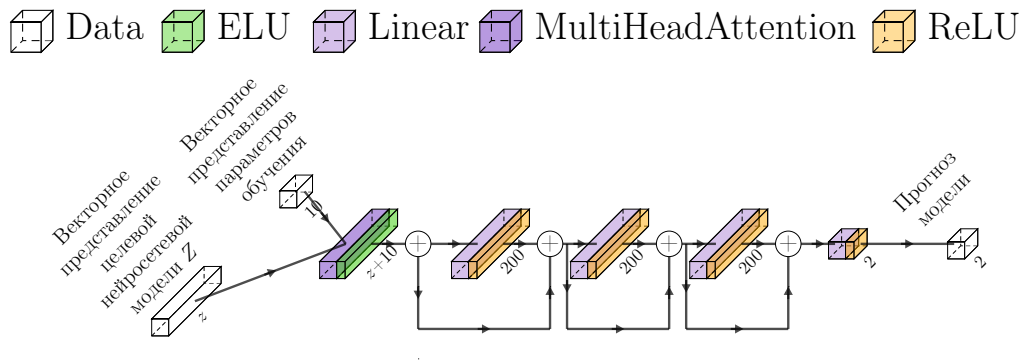


Рис. 3.7. Архитектура модели для формирования прогноза параметров обучения

Перед подачей на вход Агрегатору векторные представления объединяются посредством конкатенации в вектор длины $z + 10$. Полученный вектор обрабатывается последовательностью нейросетевых слоев, включающей один слой с механизмом внимания и пять полносвязных слоев. Скрытое состояние слоя внимания имеет размерность $z + 10$. Извлеченные данным слоем признаки обрабатываются последовательностью из пяти полносвязных слоев, формирующих прогноз качества целевой нейросетевой модели. Первые четыре слоя содержат по 200 нейронов и используют функцию активации ELU. Последний слой состоит из двух нейронов, соответствующих количеству предсказываемых величин, и формирует выходной вектор V. Первый элемент вектора V интерпретируется как ожидаемая ошибка модели, второй как прогноз времени обучения на одной эпохи.

Между слоем с механизмом внимания и последовательностью из четырех первых полносвязных слоев реализовано остаточное соединение (residual connection) [58]. Для повышения обобщающей способности модели к полносвязные слоям применяется прореживание (Dropout) [120].

3.2.3. Обучение нейросетевых моделей

Формирование обучающей выборки

Для обучения описанной выше модели используется упорядоченный набор из n четырехэлементных кортежей $M = \{(\widehat{L}_i, R_i, G_i, V_i)\}_{i=1}^{n}$, где $\widehat{L}_i$ —

матрица слоев, R_i — матрица связей между слоями, G_i — вектор параметров обучения, V_i — вектор значений качества модели. Каждый элемент $(\widehat{L}_i, R_i, G_i, V_i)$ описывает одну целевую нейросетевую модель, используемую для обучения. Для каждого элемента из набора M предполагается, что была проведена процедура обучения целевой модели, задаваемой матрицами $\widehat{L}_i$ и R_i, с использованием параметров обучения G_i. По результатам обучения и последующего тестирования на валидационной выборке был получен вектор оценки V_i.

Перед использованием набора M для обучения модели производится предварительная обработка, включающая следующие этапы: нормализация, очистка и формирование выборок. Векторы качества моделей V_i подвергаются процедуре нормализации, включающей два последовательных преобразования. На первом этапе к каждому элементу вектора применяется логарифмическое преобразование. С целью предотвращения неопределенностей, возникающих при наличии нулевых значений, к каждому элементу предварительно прибавляется единица. На втором этапе осуществляется z-нормализация. В результате данной процедуры логарифмированные значения показателей качества модели преобразуются таким образом, что математическое ожидание всех показателей было равно нулю, а стандартное отклонение было равно единице. Нормализованное значение $v_j \in V_i$ вычисляется следующим образом:

$$\hat{v_{j}}_i = \frac{\log(v_j + 1) - \mu_j}{\sigma_j}, 1 \leqslant j \leqslant |V_i|, \tag{3.63}$$

где v_j — исходное значение показателя качества, μ_j и σ_j — среднее значение и стандартное отклонение, вычисленные по всем значениям данного показателя в наборе M.

$$\mu = \frac{1}{n}\sum_{j=1}^{n}\log(v_j + 1), \sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(\log(v_j + 1) - \mu\right)^2}. \tag{3.64}$$

Входные данные представляют собой кортеж, включающий графовое представление целевой модели и вектор параметров обучения G_i. Описание целевой модели передается в виде нормализованной матрицы слоев $\widehat{L}_i$ и матрицы связей R_i. Выходными полагается кортеж, содержащий подаваемую на вход целевую нейросетевую модель и вектор параметров качества V_i. Формально обучающая выборка может быть представлена следующим образом:

$$D = \{\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \mid Y_i = (\widehat{L}_i, R_i, V_i),\ X_i = (\widehat{L}_i, R_i, G_i),\ 1 \leqslant i \leqslant n\}. \qquad (3.65)$$

Предполагается, что в процессе работы нейросетевая модель, получая на вход элементы входных данных X_i, формирует выходные данные в виде кортежа $Y_i^* = (\widehat{L}_i^*, R_i^*, V_i^*)$, включающего декодированную матрицу слоев $\widehat{L}_i^*$, декодированную матрицу связей R_i^* и прогнозируемые показатели качества V_i^*.

Вычисление ошибки

Ошибка $\boldsymbol{E}$ представляет собой составную величину и определяется как взвешенная сумма нескольких компонент, каждая из которых отвечает за оценку отклонения нейросетевой модели по одному из продуцируемых значений. Пусть $\mathcal{E} = \{E_i\}_{i=1}^{err}$ — набор ошибок, где E_i обозначает значение i-й компонентной ошибки, err — общее количество таких компонент. Каждой компоненте сопоставляется весовой коэффициент, отражающий ее вклад в итоговую ошибку. Совокупность весов задается вектором $W = \{w_i\}_{i=1}^{err}$, $w_i \in \mathbb{R}$. Суммарная ошибка нейросетевой модели определяется следующим образом:

$$\boldsymbol{E} = \sum_{i=1}^{err} w_i \cdot E_i. \qquad (3.66)$$

Составные части общей ошибки можно разделить на три группы: вероятностные ошибки, ошибки классификации и ошибки регрессии. К вероятностным ошибкам относится ошибка наличия связи между слоями E_{edge}. К ошибкам классификации относятся прогноз типа слоя E_{layer} и прогноз типа

активации слоя E_{activate}. К ошибкам регрессии относятся прогноз параметров слоев E_{params}, прогноз ошибки E_{error} и времени обучения E_{time} целевой нейросетевой модели.

Для ошибок классификации используется кросс-энтропия [48]. В контексте рассматриваемой архитектуры ошибка прогноза слоя E_{layer} определяется как значение функции кросс-энтропии, вычисленной между истинными и предсказанными распределениями вероятностей по классам слоев. Согласно формуле (3.61) истинное и предсказанное распределения расположены в первых c столбцах матриц $\widehat{L}$ и $\widehat{L}^*$. Аналогично, ошибка прогноза функции активации слоя E_{activate} определяется как значение функции кросс-энтропии, вычисленной между истинными и предсказанными распределениями вероятностей по типам функции потерь, которые расположены в последних a столбцах $\widehat{L}$ и $\widehat{L}^*$.

$$\begin{aligned} E_{\text{layer}} &= -\frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{c}\widehat{L}(i,j)\log\widehat{L}^*(i,j), \\ E_{\text{activate}} &= -\frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=c+3}^{c+a+2}\widehat{L}(i,j)\log\widehat{L}^*(i,j). \end{aligned} \tag{3.67}$$

Ошибка прогноза наличия связей между слоями определяется с помощью бинарной кросс-энтропии (binary cross entropy, BCE). Данную ошибку можно интерпретировать как меру расхождения между предсказанными и истинными вероятностями участия каждого слоя в конкретной связи. Истинные и предсказанные вероятности представляют собой матрицы $\widehat{R}$ и $\widehat{R}^*$. Предполагается, что истинные вероятности были получены после one-hot кодирования индекса слоя участника связи (см. формулу 3.60), тогда как предсказанные были сформированы нейросетевой моделью. Формально ошибка прогноза наличия связей может быть представлена следующим образом:

$$\begin{aligned} E_{\text{edge}} = -\frac{1}{\lambda}\sum_{i=1}^{\lambda}\sum_{j=1}^{2\ell}\Big(&\widehat{R}(i,j)\log\widehat{R}^*(i,j)+ \\ &+(1-\widehat{R}(i,j))\log(1-\widehat{R}^*(i,j))\Big). \end{aligned} \tag{3.68}$$

Для вычисления ошибок регрессии используется функция потерь Хубера (Huber loss) [64], которая оценивает расстояние между истинными значениями и предсказанными нейросетевой моделью. В соответствии с формулой (3.61), при вычислении ошибки прогноза параметров нейросетевых слоев E_{params} истинные значения берутся из диапазона столбцов $[c+1, c+2]$ матриц $\widehat{L}$ и $\widehat{L}^*$. В случае ошибок прогноза точности E_{error} и времени выполнения E_{time} истинные значения представлены первым и вторым столбцами матрицы V, а предсказанные соответствующими столбцами матрицы V^*. Формально ошибки данной группы могут быть обозначены следующим образом:

$$\begin{gathered} E_{\text{params}} = \frac{1}{\ell}\sum_{i=1}^{\ell} \text{HuberLoss}(\text{slice}_{i,\,c+1:v}(\widehat{L}),\ \text{slice}_{i,\,c+1:v}(\widehat{L}^*)), \\ E_{\text{error}} = \frac{1}{\ell}\sum_{i=1}^{c} \text{HuberLoss}(V(i,1),\ V^*(i,1)), \\ E_{\text{time}} = \frac{1}{\ell}\sum_{i=1}^{\ell} \text{HuberLoss}(V(i,2),\ V^*(i,2)), \quad v = c+2, \end{gathered} \tag{3.69}$$

где функция потерь Хубера $\text{HuberLoss} : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ имеет следующий вид:

$$\begin{gathered} \text{HuberLoss}(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{q} h(x_k, y_k), \\ h(x,y) = \begin{cases} \frac{1}{2}(x-y)^2, |x-y| \le \delta, \\ \delta \cdot (|x-y| - \frac{1}{2}\delta), \end{cases}, \\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^q, \quad \delta > 0. \end{gathered} \tag{3.70}$$

3.3. Выводы по главе 3

В данной главе представлены новая функция потерь MPDE (Mean Profile Distance Error) и новый метод прогнозирования ошибки и времени обучения tsGAP (time-series Graph-Attention Perfomance Prediction model), пред-

назначенные для оценки точности нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. Предложенные методы могут использоваться в качестве инструментальных средств разработки различных этапов жизненного цикла нейросетевых моделей. Функция потерь MPDE повышает точность нейросетевых моделей восстановления за счет учета поведенческих особенностей подпоследовательностей в процессе обучения. Метод tsGAP может быть использован в качестве инструментального средства предварительной оценки на этапе выбора модели.

Функция потерь MPDE предназначена для обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. В отличие от классических функций потерь, MPDE позволяет модели минимизировать поведенческие различия подпоследовательностей в процессе обучения. Формально MPDE определяется как взвешенная сумма евклидовых расстояний между исходными и нормализованными окнами подпоследовательностей. Градиент MPDE включает три следующих компонента: производная расстояния между окнами восстановленного и истинного временного ряда, производная расстояния между их z-нормализованными окнами, и производные, связанные с нормализацией. При вычислении градиента учитывается не только разница между значениями, но и то, как изменение одного значения влияет на нормализованные значения всех остальных точек в окне. Поскольку прямое вычисление MPDE обладает высокой вычислительной сложностью, в работе был предложен параллельный алгоритм ее вычисления. Разработанный алгоритм обеспечивает возможность интеграции предложенной функции потерь в современные фреймворки глубокого обучения.

Метод tsGAP используется для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. На вход метода поступают целевые нейросетевые модели восстановления потоковых данных и вектор параметров их обучения. Выходом метода полагаются вектора, содержащие прогнозируемую ошибку и время обучения на одной эпохе. Целевые нейросетевые

модели подаются на вход в виде ориентированного ациклического графа и вектора параметров обучения. Прогноз формируется с помощью нейронной сети, состоящей из трех компонентов. Автоэнкодер кодирует архитектуру нейросетевой модели в векторное представление. Энкодер параметров обучения формирует векторное представление гиперпараметров. Агрегатор объединяет полученные векторные представления и на их основе генерирует итоговый прогноз. Ошибка рассчитывается как взвешенная сумма компонент трех типов: вероятностных, классификационных и регрессионных. К вероятностным компонентам ошибки относится оценка корректности предсказания наличия связей между слоями модели. Классификационные ошибки оценивают расхождение между истинными и предсказанными нейросетевой моделью типами слоев и функциями активации. Регрессионные ошибки оценивают численные параметры слоев модели и прогнозируемые характеристики качества целевой модели.

Результаты, приведенные в этой главе, опубликованы в статьях [10, 11].

Глава 4. Вычислительные эксперименты

В данной главе представлены вычислительные эксперименты, исследующие эффективность предложенных методов, нейросетевых моделей и функции потерь. В разделе 4.1 резюмированы используемые в экспериментах наборы данных, их категории, сценарии формирования пропусков и метрики для оценки качества восстановления потоковых данных, представленных в форме временных рядов. Разделы 4.2, 4.3, 4.4 и 4.5 посвящены сравнительному анализу методов восстановления SANNI, SAETI, функции потерь MPDE и метода tsGAP с передовыми аналогами соответственно.

4.1. Наборы данных и сценарии формирования пропусков

В данном исследовании для оценки ошибки восстановления пропущенных значений используется корень из среднеквадратичной ошибки (Root Mean Square Error, RMSE). Данная метрика является одной из наиболее распространенных в подобных исследованиях [100] и формально определяется следующим образом:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{4.1}$$

где y_i — фактическое значение, $\hat{y}_i$ — предсказанное значение, n — количество наблюдений.

Наборы данных

В ходе экспериментов использовались наборы данных, представленные в табл. 4.1. Временные ряды можно разделять не только по числовым характеристикам, таким как дисперсия, автокорреляция и др., но и по семантической природе процессов, отражаемых во временных рядах. Одним из ключевых факторов, определяющих способ разделения на категории в дан-

ном исследовании, является наличие повторяющихся действий или закономерностей, отражающих поведение объектов или субъектов во времени. Подобные закономерности создают шаблоны, которые могут быть полезны для методов восстановления пропущенных данных, поскольку они позволяют предсказывать недостающие значения на основе уже наблюдаемых повторяющихся структур.

Табл. 4.1. Наборы данных, используемые в экспериментах

№	Набор данных	Длина, $n \times 10^3$	Количество, измерений d	Предметная область
Категория А: Активности				
1.	Electricity	5	9	Потребление электроэнергии в нескольких домашних хозяйствах Франции
2.	Madrid	25	10	Автоматическая регистрация транспортных средств в городе Мадрид
3.	NREL	8.7	9	Потребление электроэнергии исследовательской лаборатории в США
4.	PAMAP	50	10	Показания носимого датчика во время активности человека
5.	WalkRun	37	11	
Категория Б: Сезонность и/или цикличность				
6.	BAFU	50	10	Сброс воды в реках Швейцарии
7.	Climate	5	10	Погода в различных локациях Северной Америки
8.	MAREL	50	10	Качество морской воды в Ла-Манше
9.	MeteoSwiss	10	10	Погода в городах Швейцарии
10.	Saaleaue	23	14	Погодные данные в Германии
Категория В: Стохастичность				
11.	BTC	2.5	12	Курсы криптовалют
12.	Soccer	500	10	Данные с носимых сенсоров футболистов

Рассмотрим категории временных рядов, используемых в данном исследовании. Категория А включает ряды, отражающие деятельность одного или нескольких субъектов, выполняющих повторяющиеся действия и демонстрирующих определенные закономерности. Категория Б охватывает сезонные и циклические временные ряды, где колебания имеют соответственно фиксированные и переменные периоды. Категория В представляет стохастические временные ряды, описывающие недетерминированное поведение субъектов без выраженной сезонности или цикличности. Предполагается, что категория А является целевой для данной работы, поскольку временные ряды этой категории содержат повторяющиеся действия субъектов, которые можно рассматривать как информативные шаблоны.

В категорию А входят следующие ряды. Набор данных *Madrid* [81] содержит временные ряды транспортного потока, собранные с автоматических устройств регистрации транспортных средств (Automatic Vehicle Registration, AVR) на дорогах и автомагистралях Мадрида в период с 2014

по 2016 год. Шаблоны активностей в таком ряду могут быть связаны с повторяющимися потоками транспорта в рабочие и выходные дни. *NREL* [114] содержит почасовые данные о потреблении электроэнергии различными инженерными системами Национальной лаборатории возобновляемой энергии (National Renewable Energy Laboratory, NREL) в США. Потенциальные шаблоны могут отражать регулярное изменение активности систем по времени суток или в разные дни недели. *PAMAP* [110] содержит данные мониторинга физической активности одного спортсмена, выполняющего шесть различных видов активности. Шаблоны могут соответствовать повторяющимся циклам каждой активности, например ходьба, бег, приседания. *WalkRun* [110] отражает измерения акселерометра, гироскопа и магнитометра при чередовании ходьбы и бега. Шаблоны проявляются как регулярные циклы движений конечностей в двух активностях. Набор данных *Electricity* [130] содержит потребление электроэнергии домохозяйствами. Возможные шаблоны в этом ряду связаны с повторяющимися рабочими и выходными днями.

Категория Б включает следующие ряды. *BAFU* [27] содержит данные об объеме стока воды в реках Швейцарии. *Climate* [93] представляет собой ежемесячные климатические измерения, собранные в различных регионах Северной Америки в период с 1990 по 2002 год. *MAREL* [84] включает химические и биологические характеристики морской воды в Ла-Манше. *MeteoSwiss* [99] фиксирует различные метеорологические измерения в городах Швейцарии. *Saaleaue* [138] содержит метеоданные, включая температуру, влажность и концентрацию CO_2, собранные в Германии с 2009 по 2016 год.

Категория включает следующие ряды. *BTC* [60] включает исторические данные о курсах криптовалют Bitcoin, Ethereum и Monero. Временной ряд характеризуется высокой волатильностью и низкой предсказуемостью. *Soccer* [102] содержит данные с сенсоров игроков во время футбольного матча с частотой измерений 200 Гц, что создает большое количество событий с высокой вариативностью, практически без повторяющихся шаблонов.

Сценарии формирования пропусков

(a) Blackout (b) MCAR (c) Disjoint (d) Overlap

Рис. 4.1. Сценарии формирования пропусков

Для проведения вычислительных экспериментов в данные целенаправленно вносились пропуски в соответствии с четырьмя сценариями, описанными в работе [76]: Blackout, MCAR, Disjoint и Overlap (см. рис. 4.1).

В сценарии Blackout последние 100 точек каждого измерения временного ряда отсутствуют. Данный сценарий представляет собой серьезную задачу для методов восстановления пропусков, поскольку требует оценки значений в условиях полного отсутствия наблюдений в продолжительном интервале [76].

Сценарий MCAR (Missing Completely at Random) предполагает случайное исключение блоков длиной 10 значений в произвольно выбранных измерениях. Пропуски добавляются до достижения общей доли в 10% от общего числа значений. Выбор блоков осуществляется с использованием генератора случайных чисел. Для обеспечения воспроизводимости во всех экспериментах используется генератор с фиксированным зерном (seed).

В сценарии Disjoint в каждом измерении временного ряда формируется один пропущенный блок длиной $\lceil n/d \rceil$, где n — длина ряда, а d — количество измерений. Начальная позиция блока в i-м измерении определяется как $i \cdot \lceil n/d \rceil$. Сценарий Overlap представляет собой модификацию Disjoint с удвоенной длиной пропущенного блока и при сохранении той же начальной позиции В результате формируются блоки пропусков, частично перекрывающиеся между различными измерениями.

4.2. Нейросетевой метод SANNI

4.2.1. Описание экспериментов

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты, в которых оценивалась точность восстановления предложенного метода на реальных и синтетических наборах данных. Выполнено сравнение метода SANNI с конкурентами, рассмотренными выше в разделах 1.3.1 и 1.4.1.

Для экспериментов были выбраны следующие нейросетевые методы: NAOMI [92], BRITS [30], GP-VAE [49], M-RNN [153], SAITS и Transformer [44]. Для обеспечения корректности и сопоставимости результатов применялись оригинальные реализации указанных методов, предоставленные их авторами.

В ходе экспериментов использовались следующие аналитические алгоритмы: CDRec [75], DynaMMo [88], ORBITS [73], SVT [29], ROSL [116], GROUSE [21], SoftImpute [97], SVDImpute [131] и TeNMF [98]. Для всех указанных аналитических алгоритмов использовались открытые реализации, включая исходный код, предоставленный авторами ORBITS [73]. Для обеспечения максимально возможной точности восстановления в экспериментах гиперпараметры каждого метода подбирались в соответствии с рекомендациями оригинальных работ либо с реализацией, предложенной авторами метода ORBITS [73]. Для справедливого сравнения длина подпоследовательностей во всех нейросетевых методах задавалась одинаковой.

В экспериментах использовались временные ряды, подробно описанные в разделе 4.1. Сравнение методов восстановления осуществлялось на выборках, в которых пропуски были внесены до проведения обучения и последующего восстановления. Пропуски во временные ряды добавлялись в соответствии со сценариями, изложенными в разделе 4.1. Следует отметить, что часть алгоритмов восстановления, включая GROUSE, ORBITS, SoftImpute, SVDImpute, SVT и TeNMF, не применимы в Blackout-сценарии, поскольку предполагают наличие хотя бы одного непропущенного значе-

ния в каждой многомерной точке. Данные с пропусками, сформированные согласно описанным сценариям, рассматривались в качестве тестового набора. В тех случаях, когда метод требовал разбиения на обучающий и валидационный выборки, из множества всех подпоследовательностей временного ряда случайным образом выбирались многомерные подпоследовательности в соотношении 75% и 25%.

В ходе экспериментов каждый алгоритм или метод восстанавливал пропуски, добавленные в исходные временные ряды. Затем вычислялась ошибка восстановления, которая отражает степень совпадения восстановленных данных с истинными значениями, находившимися на местах пропусков в исходных рядах. Для оценки точности восстановления пропущенных значений использовалась метрика RMSE (см. формулу 4.1).

Табл. 4.2. Оборудование, использованное в экспериментах

Характеристика	Процессор (CPU)	Видеокарта (GPU)
Производитель и серия	Intel Xeon	NVIDIA Ampere
Модель	E5-2687W v2	A100
Количество ядер	8	6 912
Тактовая частота, ГГц	3.40	1.41
Объем памяти, ГБ	16	80
Пиковая производительность (double precision), TFLOPS	0.157	9.7

Вычислительные эксперименты проводились на оборудовании Южно-Уральского государственного университета (Челябинск) [1]. В табл. 4.2 приведены характеристики аппаратных платформ, на которых были проведены вычислительные эксперименты. Параметры метода SANNI, которые использовались в вычислительных экспериментах, представлены в табл. 4.3.

4.2.2. Анализ результатов

На рис. 4.2 представлена усредненная по категориям наборов данных ошибка восстановления. Подробные результаты приведены в Приложении А (см. таблицы А.1, А.2, А.3, А.4).

Табл. 4.3. Гиперпараметры SANNI

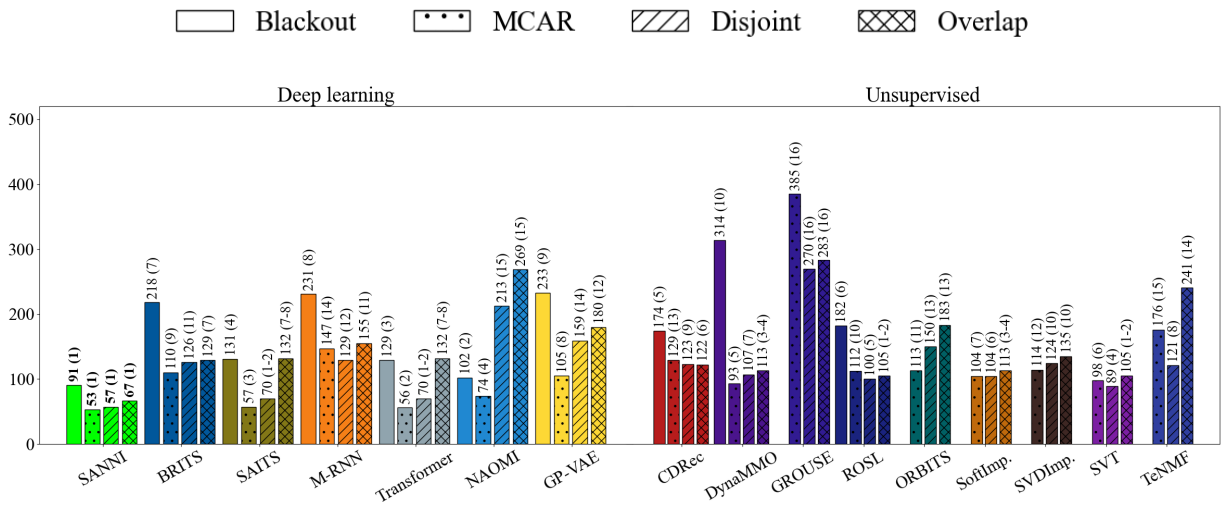
Гиперпараметр	Значение
Длина сниппета, m	200
Количество классов, K	2
Оптимизатор	Adam
Начальная скорость обучения	0.0005
Размер пакета	128
Количество эпох для Распознавателя	100
Количество эпох для Реконструктора	1000

На рис. 4.2a представлены результаты вычислительных экспериментов для категории А. Как видно из рисунка, во всех сценариях предложенный метод обеспечивает превосходство по точности над конкурентами. В среднем, для данной категории точность метода выше на 56% относительно передовых аналогов. При этом по сравнению с конкурентом, показавшим минимальное значение ошибки, преимущество составляет 17%.

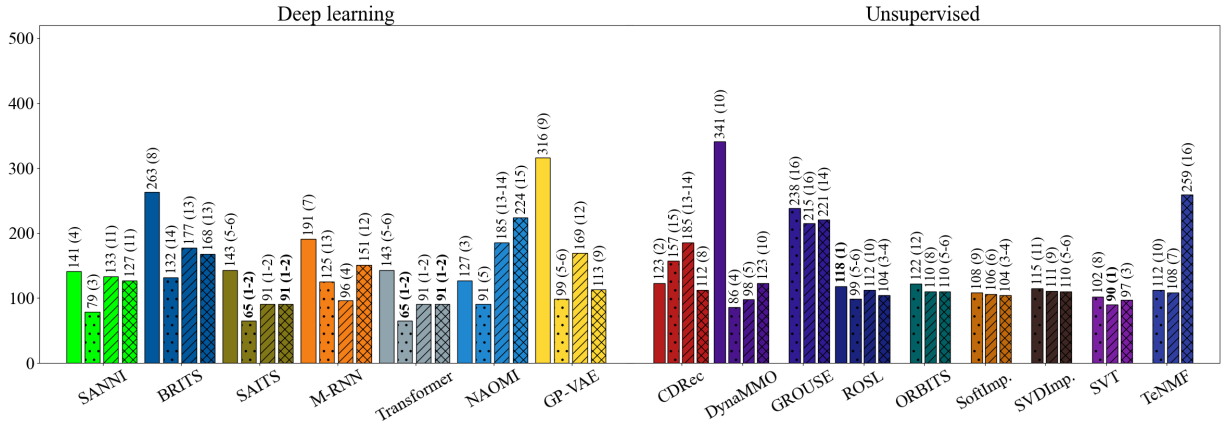
Результаты для категории Б представлены на рис. 4.2b. Для данной категории метод SANNI показывает умеренные результаты: он занимает 4-е место в сценарии Blackout и 3-е в сценарии MCAR. Однако для других сценариев (Disjoint и Overlap) точность SANNI значительно уступает другим методам. В среднем, для данной категории точность SANNI выше на 20% по сравнению с другими методами восстановления, тогда как относительно наилучшего конкурента наблюдается снижение точности на 31%.

Для категории В (см. рис. 4.2c) наблюдается аналогичная тенденция, как и для категории Б. Метод SANNI демонстрирует наилучшую точность в сценарии Blackout, занимает 4-ю позицию в сценарии MCAR, но значительно проигрывает другим подходам в сценариях Disjoint и Overlap. В среднем, для данной категории точность метода выше на 31% относительно других современных алгоритмов, при этом по сравнению с наилучшим конкурентом наблюдается снижение точности на 51%.

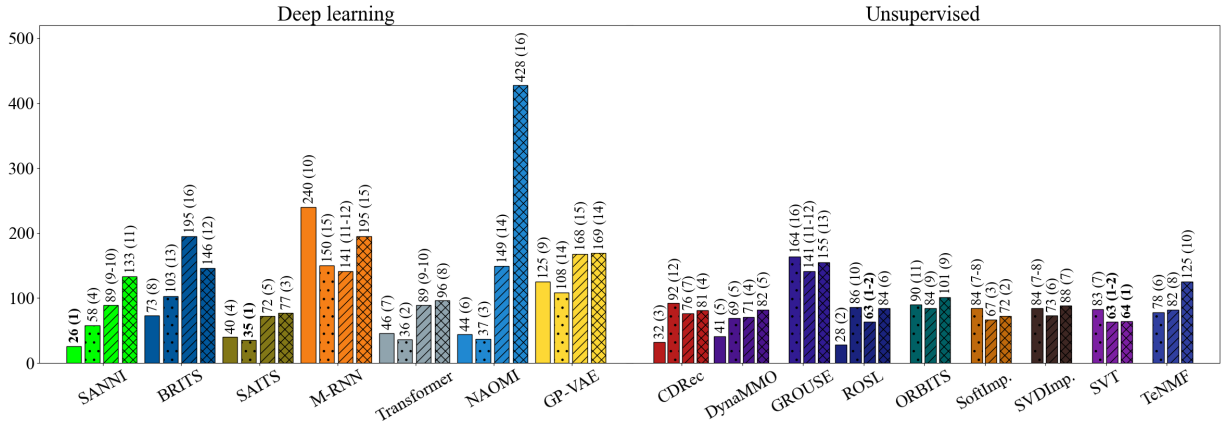
Полученные результаты указывают на влияние категории временных рядов на эффективность метода SANNI. Наибольшая точность достигается на рядах с повторяющимися активностями, поскольку в таких рядах метод



(a) Категория А: Активности



(b) Категория Б: Сезонности и циклы



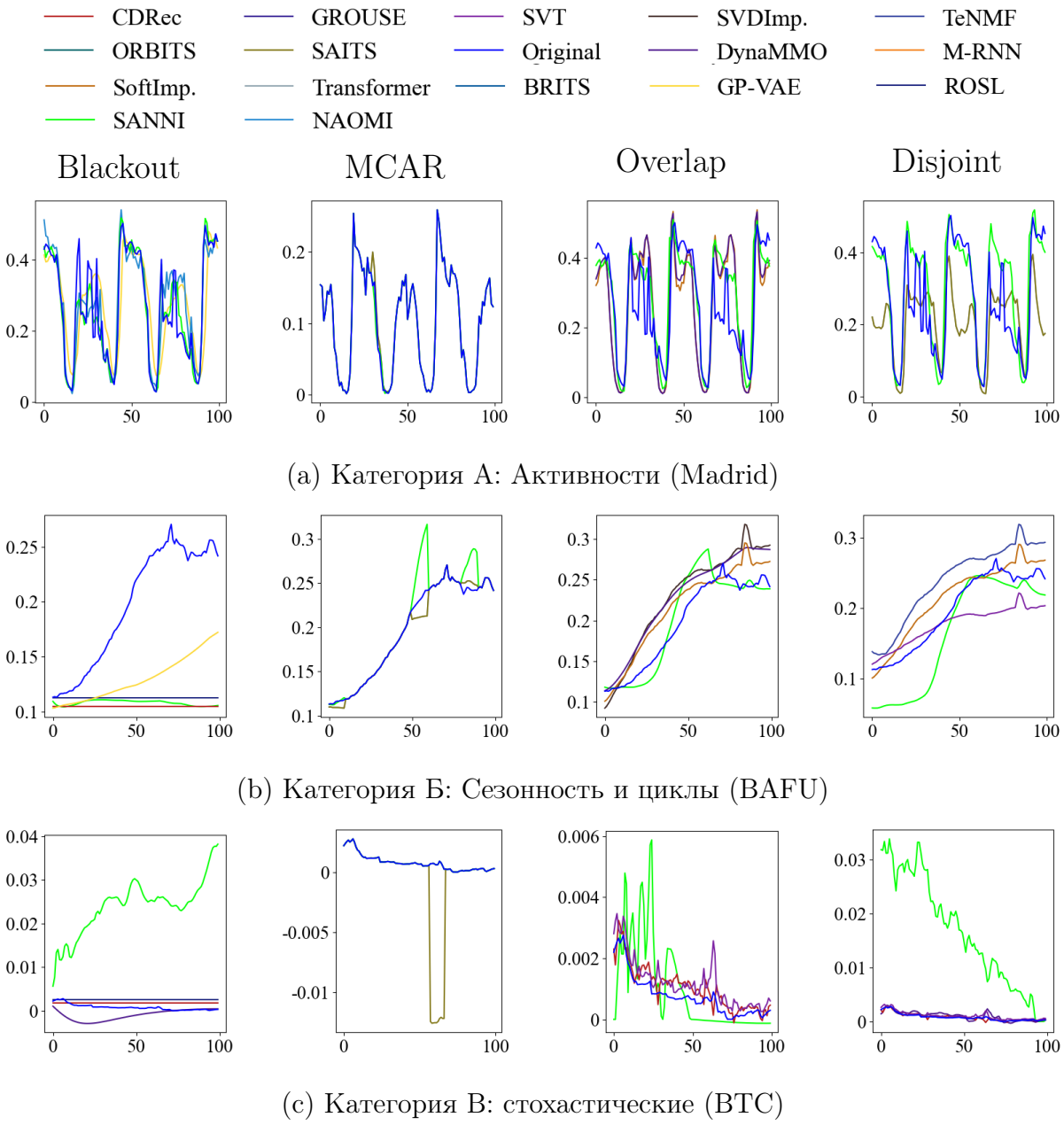
(c) Категория В: Стохастические

Рис. 4.2. Сравнение ошибки восстановления, RMSE $\times 10^{-3}$

способен выявлять шаблоны и использовать их для повышения точности восстановления отсутствующих значений. На рядах с выраженной сезонностью или стохастической природой наблюдается снижение точности относительно наилучших методов, поскольку в этих рядах отсутствуют четко выраженные повторяющиеся шаблоны. Однако сохранение превосходства над средним уровнем аналогов указывает на общую устойчивость и способность модели частично адаптироваться к отсутствию полезных шаблонов.

Наибольшее преимущество предложенный метод демонстрирует в сценарии Blackout. В остальных сценариях аналогичные подходы могут использовать данные из других измерений для восстановления пропусков, тогда как SANNI оперирует исключительно данными, которые предшествуют пропускам, и не анализирует показания других измерений, полученные в текущий или «будущие» моменты времени. В условиях Blackout, когда данные из других измерений отсутствуют, особенности реализации метода SANNI позволяют максимально эффективно использовать доступные исторические наблюдения, обеспечивая высокую точность восстановления относительно конкурентов. Ограничение SANNI по использованию только исторических данных текущего измерения совпадает с условиями работы в режиме онлайн, что делает метод особенно пригодным для практических приложений, где доступ к будущей или синхронной информации ограничен.

На рис. 4.3 приведены примеры восстановления пропущенных значений. Показаны фрагменты из 100 точек истинных и восстановленных временных рядов, полученные с использованием пяти методов, показавших наибольшую точность. Для наглядности выбраны наборы данных Madrid, BAFU и BTC, представляющие категории А, Б и В соответственно. Для каждого набора демонстрируются результаты по всем сценариям восстановления. Из рисунка видно, что метод SANNI обеспечивает высокую точность восстановления для категории А. Для категорий Б и В метод демонстрирует результаты, приближенные к показателям других методов восстановления, оставаясь на уровне конкурентов.



(a) Категория А: Активности (Madrid)

(b) Категория Б: Сезонность и циклы (BAFU)

(c) Категория В: стохастические (BTC)

Рис. 4.3. Примеры восстановления наборов данных из разных категорий

На рис. 4.4 представлены результаты экспериментов, содержащие производительность сравниваемых нейросетевых методов. Из рисунка видно, что модели SAITS и Transformer существенно превосходят остальные подходы по скорости обучения. Подобное преимущество объясняется тем, что в данных нейросетевых моделях не используются рекуррентные слои. Вместо этого для анализа последовательных данных они используют механизм самовнимания (self-attention) [132], который позволяет эффективно моде-

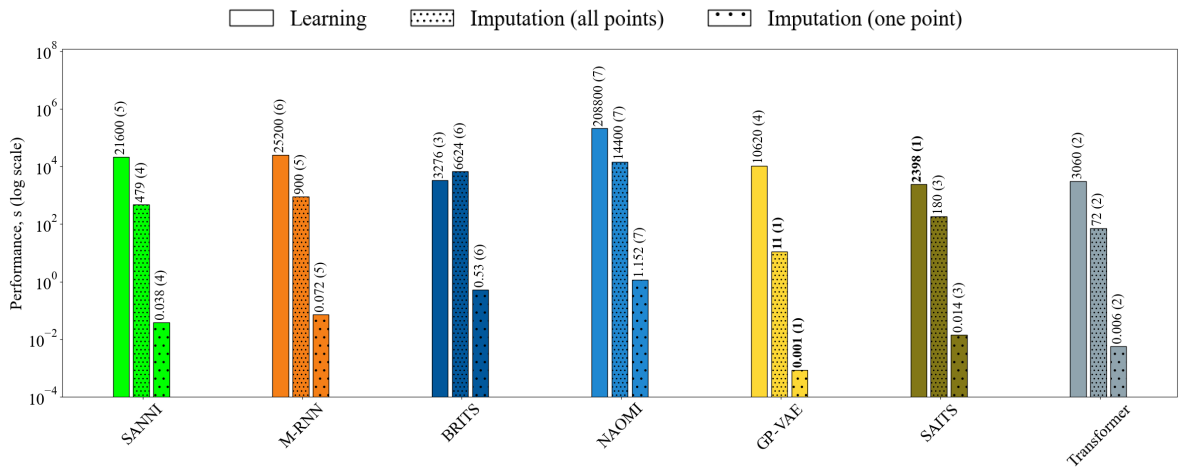


Рис. 4.4. Сравнение производительности нейросетевых методов

лировать зависимости между элементами последовательности без итеративного прохода по временным шагам. Нейросетевая модель GP-VAE, также не содержащая рекуррентных компонентов, демонстрирует наилучшие результаты при восстановлении как отдельных пропущенных значений, так и непрерывных интервалов с пропусками.

Метод SANNI показывает более низкую производительность по сравнению с указанными методами. В связи с дополнительными этапами обработки, общее время обучения метода возрастает. При этом метод использует нейросетевые модели, включающие рекуррентные слои, замедляющие процесс восстановления. Тем не менее, несмотря на вычислительные издержки, SANNI превосходит по производительности восстановления другие рекуррентные методы, такие как BRITS и M-RNN.

Рассмотрим применимость метода SANNI для восстановления пропусков в режиме онлайн. В системах автоматизации зданий (Building Automation, BA) и промышленных процессах (Process Automation, PA) типичная частота обновления сенсоров составляет соответственно 10 с и 100 мс [96]. К системам BA относятся различные управляющие подсистемы зданий, включая противопожарную безопасность, освещение, отопление, водоснабжение, кондиционирование и др. [167]. Область применения PA охватывает химическую, фармацевтическую, горнодобывающую, нефтегазовую, металлургическую и ряд других отраслей [155]. Поскольку SANNI восста-

навливает одну точку за 38 мс., он удовлетворяет требованиям работы в режиме реального времени.

4.3. Нейросетевой метод SAETI

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты на оборудовании Лаборатории суперкомпьютерного моделирования ЮУрГУ [1].

4.3.1. Описание экспериментов

В рамках исследования были проведены две серии экспериментов. В первой серии экспериментов выполнялось сравнение предложенного метода с существующими аналогами. Во второй серии анализировалось влияние шаблонов на точность восстановления предлагаемого метода.

В первой серии экспериментов оценка предложенного метода и его сравнение с аналогами проводились с использованием временных рядов, резюмированных в табл. 4.1. Для экспериментов были выбраны наборы данных категорий А и Б. Формирование пропусков многомерного временного ряда осуществлялось в соответствии со сценарием MCAR (Missing Completely at Random) [76]. Для оценки точности восстановления в данной работе используется мера корня из среднеквадратичной ошибки RMSE (Root Mean Square Error) (см. формулу 4.1).

Сравнение метода SAETI проводилось с конкурентами, рассмотренными ранее в разделах 1.3.1 и 1.4.1. Из числа нейросетевых методов восстановления для сравнения были выбраны следующие методы: NAOMI [92], BRITS [30], GP-VAE[49], M-RNN [153], SAITS и TRANSFORM [44]. Из числа аналитических были использованы следующие алгоритмы: CDRec [75], DynaMMo [88], ORBITS [73], ROSL [116], GROUSE [21], SoftImpute [97], SVDImpute [131], SVT [29] и TeNMF [98]. Для проведения вычислительных экспериментов были установлены следующие параметры SAETI: размер входной подпоследовательности $m = 200$, количество активностей $K = 2$,

доля подпоследовательностей ряда с пропусками $\alpha = 0.5$, доля обнуляемых элементов при подготовке обучающей выборки Реконструктора $p = 0.25$.

Во второй серии экспериментов проводилась оценка влияния шаблонов на качество восстановления пропусков. Для этого осуществлялось сравнение предложенного метода SAETI с аналогичным методом AETI, в котором, в отличие от SAETI, шаблоны не подавались на вход нейросетевой модели Реконструктор, а нейросетевая модель Распознаватель отсутствовала. Эксперименты проводились при различном объеме искусственно добавленных пропусков в данных. Объем пропусков варьировался в диапазоне от 5% до 95%. На вход обоих методов подавались подпоследовательности одинаковой длины. Значения всех гиперпараметров (включая размер входного окна, размерность скрытых представлений и параметры обучения) оставались одинаковыми для обоих методов.

4.3.2. Анализ результатов

Точность восстановления

Таблица 4.4 содержит результаты сравнения точности восстановления метода SAETI с различными методами. Методы сгруппированы построчно на две категории: аналитические алгоритмы и нейросетевые модели. По столбцам приведены результаты для наборов данных категорий А и Б. В каждой ячейке указан показатель точности восстановления (RMSE) вместе с рангом метода в скобках, определенном относительно всех конкурентов. Для каждой категории, а также для всех наборов данных в целом, приведены столбцы со средними значениями RMSE. Лучшие результаты среди всех методов выделены полужирным шрифтом.

Из таблицы видно, что в категории Б метод BRITS демонстрирует наилучшую среднюю точность восстановления как среди аналитических, так и среди нейросетевых аналогов. При этом вторую позицию в данной группе занимает предложенный метод SAETI, опережающий методы SAITS и Transformer. Для данной категории точность SAETI превышает средний

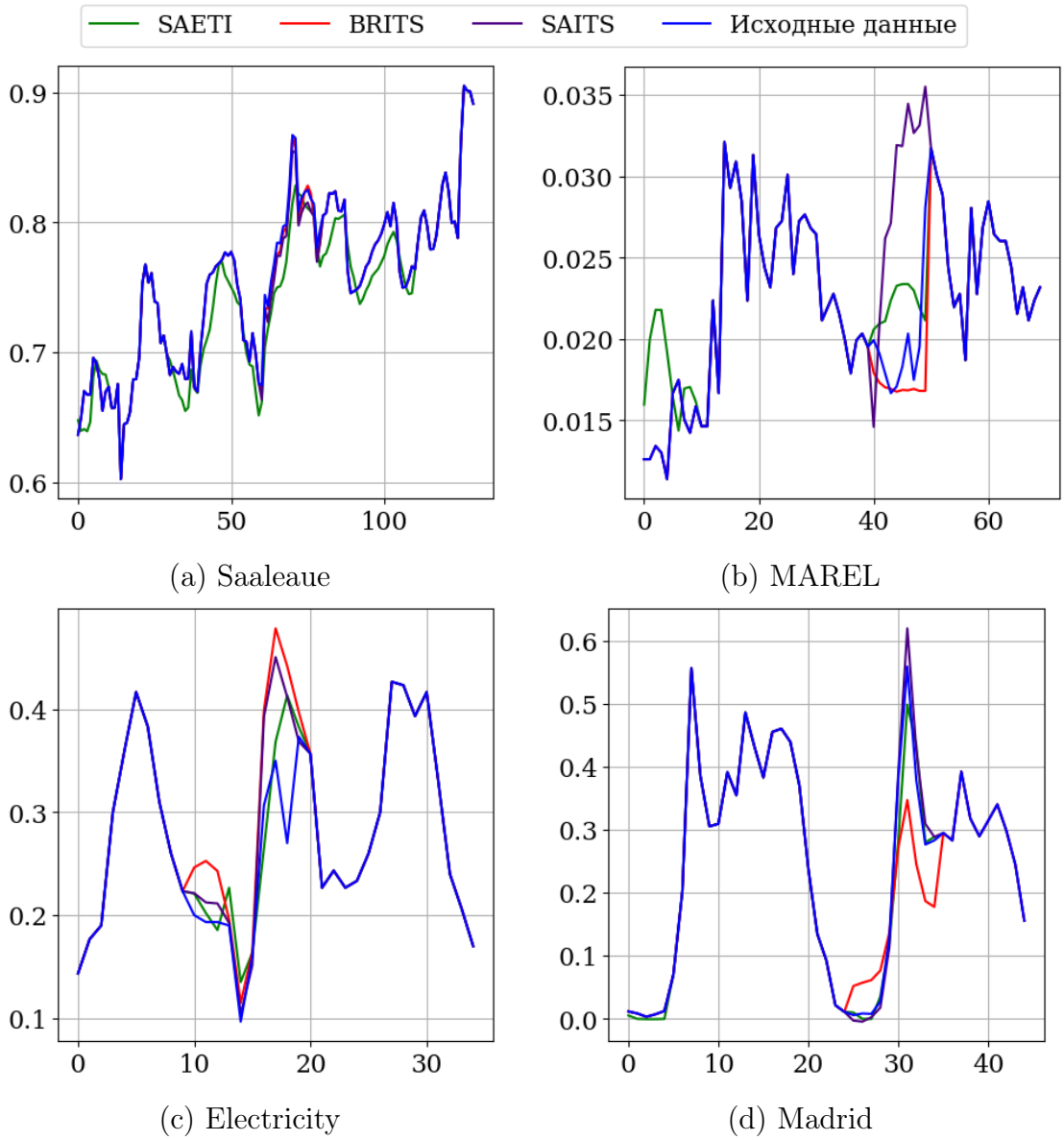
Табл. 4.4. Сравнение ошибки восстановления метода SAETI с аналогами, RMSE$\times 10^{-3}$

Методы		Наборы данных											
		Категория А				Ср. ошибка	Категория Б					Ср. ошибка	Ср. ошибка
Тип	Название	Electricity	Madrid	Soccer	WalkRun		BAFU	Climate	MAREL	MeteoSwiss	Saaleaue		
Аналитические	CDRec	107(13)	103(14)	125(13)	177(15)	128(14)	75(13)	156.2(12)	358(16)	78(9)	119(14)	157(15)	144(14)
	DynaMMo	101(8)	71(6)	79(6)	127(6)	94(6)	39(5)	142(8)	146(6)	73(5)	85.3(7)	97(7)	96(6)
	GROUSE	134(15)	280(16)	163(15)	163(12)	185(16)	206(16)	226(16)	214(13)	191(16)	155(16)	198(16)	192(16)
	ORBITS	105(10)	90(11)	113.5(12)	174(14)	120.6(13)	109(15)	166(14)	192(12)	85(11)	104(10)	131(13)	126(13)
	ROSL	111(14)	99(13)	105(7)	135(8)	113(10)	60(9)	179(15)	168(8)	101(13)	85(6)	119(10)	116(10)
	SoftImpute	103(9)	76(8)	106(8)	156(10)	110(9)	65(11)	152(10)	188(11)	80(10)	100(9)	117(9)	114(9)
	SVDImpute	105.2(11)	80(10)	110(10)	161(11)	114(11)	62(10)	156(11)	215(14)	77(7)	107(12)	123(12)	119(11)
	SVT	95(7)	80(9)	108(9)	154(9)	109(8)	74(12)	143(9)	180(10)	78(8)	75(5)	110(8)	110(8)
	TeNMF	106(12)	91(12)	113(11)	168(13)	120(12)	49(8)	163(13)	216(15)	77(6)	106(11)	122(11)	121(12)
Нейросетевые	BRITS	**57(1)**	35(2)	20(5)	68(4)	45(2)	17(4)	52(2)	**77(1)**	52(2)	**54(1)**	**50(1)**	48(2)
	GP-VAE	74(4)	73(7)	142(14)	123(5)	103(7)	46(7)	**15(1)**	172(9)	129(14)	92(8)	91(6)	96.3(7)
	M-RNN	180(16)	108(15)	164(16)	227(16)	170(15)	88(14)	138(7)	161(7)	169(15)	140(15)	139(14)	153(15)
	NAOMI	75(5)	49(5)	9(2)	134(7)	67(5)	40(6)	76(6)	128(5)	88(12)	107.1(13)	88(5)	78(5)
	SAITS	73(3)	45(3-4)	14(3)	49(2-3)	45.1(3)	15.2(2)	57.8(5)	79(2-3)	65(3-4)	57(2-3)	54.4(4)	50(3)
	Transformer	81(6)	45(3-4)	20(4)	49(2-3)	49(4)	**15(1)**	57(4)	79(2-3)	65(3-4)	57(2-3)	54(3)	52(4)
	SAETI	60(2)	**31(1)**	**6(1)**	**42(1)**	**35(1)**	16(3)	53(3)	88(4)	**52(1)**	58(4)	53(2)	**45(1)**

уровень конкурентов на 50%, при этом падение точности относительно наилучшего метода составляет 6%.

В категории А предложенный метод превосходит конкурентов по точности восстановления пропущенных значений. Точность SAETI в данной категории превышает средний уровень передовых аналогов на 63%. Преимущество относительно наилучшего конкурента составляет 22%, уступая только в одном наборе данных методу BRITS.

На рис. 4.5 представлены примеры фрагментов временных рядов, восстановленных с помощью SAETI и двух лучших конкурентов, BRITS и SAITS. При восстановлении рядов Saaleaue и MAREL (см. рис. 4.5a и 4.5b), характеризующихся возрастающим трендом и отсутствием выраженных поведенческих шаблонов, метод SAETI демонстрирует меньшую точность по сравнению с конкурентами. В то же время при обработке временных рядов Electricity и Madrid (см. рис. 4.5c и 4.5d) SAETI опережает конкурентов по точности восстановления за счет использования шаблонов поведения людей в зависимости от дня недели.



(a) Saaleaue (b) MAREL

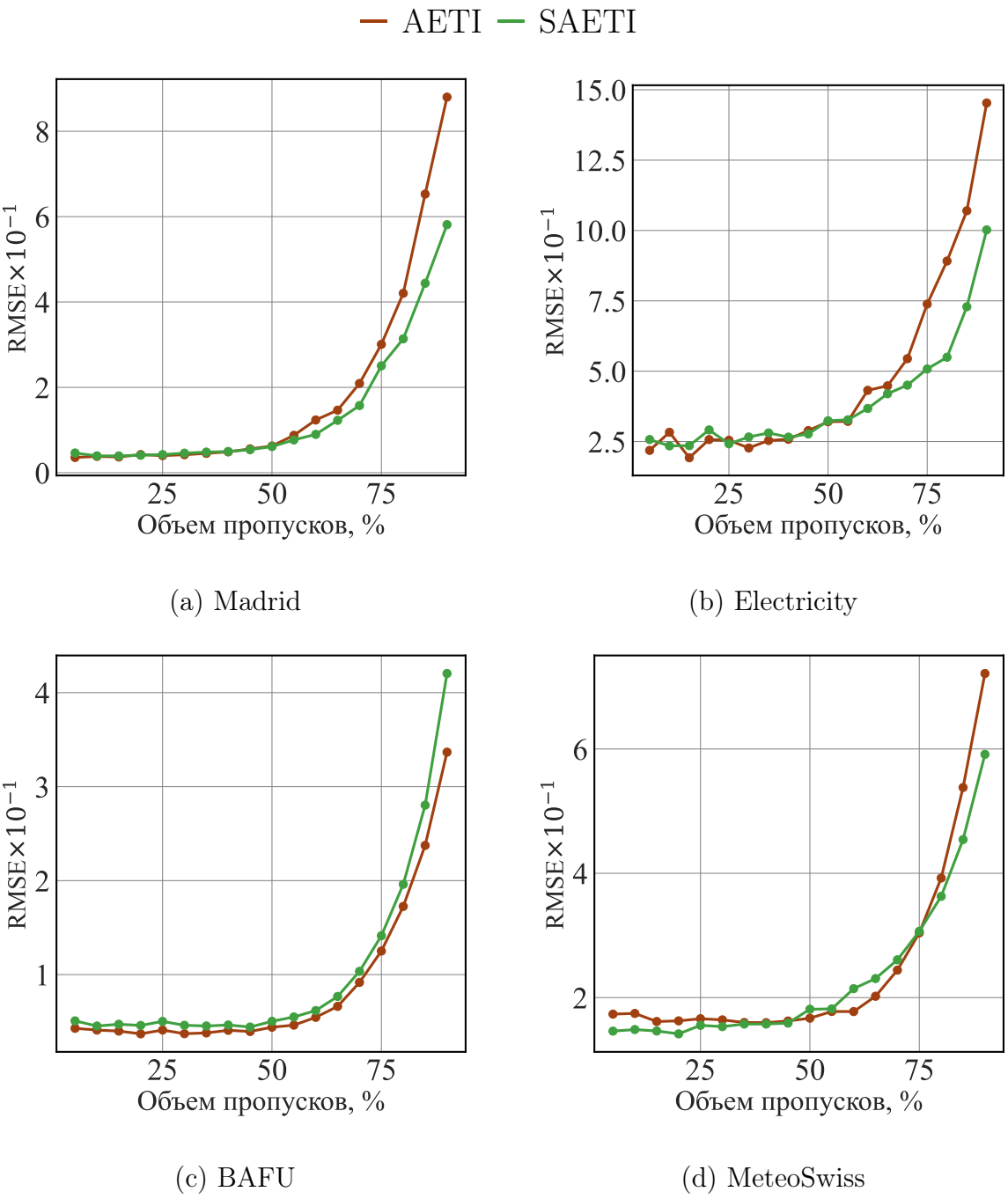
(c) Electricity (d) Madrid

Рис. 4.5. Примеры восстановления наборов данных из разных категорий

Влияние шаблонов

На рис. 4.6 представлены результаты сравнительного анализа точности методов AETI и SAETI при восстановлении пропущенных значений в данных различных наборов.

На рис. 4.6a и 4.6b показаны результаты для набора данных Madrid и Electricity, относящихся к категории A. На рис. 4.6c и 4.6d приведены результаты для наборов BAFU и MeteoSwiss, которые относятся к катего-



(a) Madrid

(b) Electricity

(c) BAFU

(d) MeteoSwiss

Рис. 4.6. Зависимость точности восстановления от объема пропусков

рии Б. Каждый рисунок представляет собой график, на котором по горизонтальной оси отображен объем пропущенных значений во входных подпоследовательностях. По вертикальной оси отложена ошибка восстановления соответствующих методов.

Для наборов данных категории А установлено, что при доле пропущенных значений до 50% значения ошибки методов AETI и SAETI остаются

сопоставимыми. При превышении данного порога наблюдается увеличение ошибки метода AETI по сравнению с методом SAETI. Отсутствие значимых различий между методами при низких долях пропусков связано с тем, что при достаточном объеме наблюдаемых данных оба метода способны эффективно восстанавливать пропущенные значения, используя только информацию во входных подпоследовательностях. В этих условиях использование шаблонов в методе SAETI не оказывает существенного влияния на результат, поскольку нейросетевая модель не получает дополнительной информации из шаблонов, уже представленной в наблюдаемых данных. При увеличении объема пропусков наблюдаемых данных становится недостаточным для корректного формирования векторного представления подпоследовательности, что приводит к увеличению ошибки восстановления метода AETI.

Для наборов данных категории Б методы AETI и SAETI демонстрируют примерно одинаковую точность восстановления вне зависимости от объема пропущенных значений. Несмотря на то, что в данной категории шаблоны могут быть выделены благодаря особенностям работы алгоритмов поиска шаблонов, содержащаяся в них информация недостаточно полезна для улучшения результатов восстановления.

4.4. Функция потерь MPDE

4.4.1. Описание экспериментов

Во время экспериментов проводилось сравнение предложенной функции потерь MPDE со следующими распространенными функциями: средней абсолютной ошибкой (Mean Absolute Error, MAE), среднеквадратичной ошибкой (Mean Squared Error, MSE), квантильной функцией потерь (Quantile Loss) [33] и функцией потерь, основанной на алгоритме трансформации временной шкалы (Dynamic Time Warping, DTW). В качестве реализации функции потерь на основе DTW используется ее модификация Soft-DTW [38], представляющая собой дифференцируемую версию DTW.

Для оценки предложенной функции потерь были проведены три серии экспериментов.

Первая серия экспериментов была направлена на оценку точности нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, обученных с использованием каждой из рассматриваемых функций потерь. Оценка функций потерь проводилась на временных рядах, приведенных в табл. 4.1. Из категории А использовались ряды Electricity, Madrid, NREL, PAMAP и WalkRun. Из категории Б были взяты ряды BAFU, Climate, MeteoSwiss и Saaleaue. Пропуски в указанных выше временных рядах формировались в соответствии со сценарием Blackout, при котором пустыми значениями считаются 100 последних точек каждого измерения многомерного временного ряда. В качестве меры оценки качества восстановления использовался корень из среднеквадратичной ошибки RMSE. RMSE вычислялся между данными, помеченными как пропущенные, и восстановленными значениями. Эксперименты проводились на различных архитектурах нейросетевых моделей, включая рекуррентные модели (BRITS [30] и SANNI [161]), автоэнкодеры (SAETI [9]) и трансформеры (SAITS [44]).

Табл. 4.5. Параметры обучения нейросетевых моделей, 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

Параметры обучения нейросетевых моделей приведены в табл. 4.5. Оптимальные значения гиперпараметров функции потерь MPDE для каждой комбинации $\langle$*Нейросетевая модель*, *Набор данных*$\rangle$ подбирались с помощью решетчатого поиска (grid search) [107]. Значения α и β подбирались в диапазоне $[0.1, 0.9]$ с шагом 0.1.

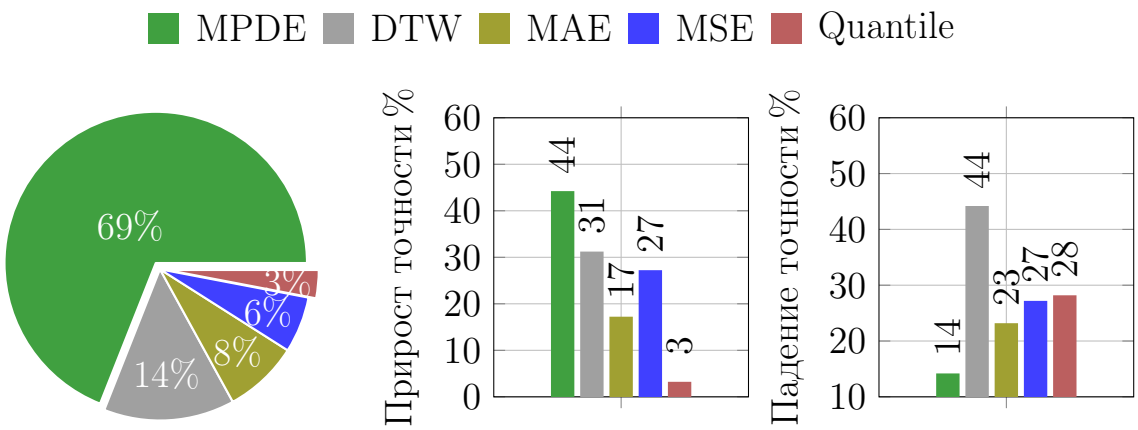
Вторая серия экспериментов изучала изменение значений различных функций потерь при переходе предсказаний модели от зашумленных данных, соответствующих ранним этапам обучения, к эталонным данным. Для моделирования постепенного приближения к эталону зашумленные данные формировались посредством линейного смешивания эталонных данных с шумом с уменьшающимся коэффициентом γ. Целью этих экспериментов было проанализировать изменение кривых функций потерь и норм градиента на различных этапах обучения.

Третья серия экспериментов была направлена на исследование производительности различных функций потерь при изменении ключевых параметров временных рядов: длины последовательности, размера батча и количества измерений. Для проведения экспериментов использовалась синтетическая одномерная подпоследовательность на основе синусоиды с добавленным шумом, имитирующим предсказания нейросетевой модели на различных этапах обучения. В ходе экспериментов фиксировались время расчета ошибки, время вычисления градиента и объем используемой видеопамяти для каждой функции потерь. Измерения проводились при варьировании длины последовательности, размера батча и количества измерений временного ряда. Цель этой серии заключалась в количественной оценке эффективности и масштабируемости различных функций потерь в зависимости от характеристик входных данных.

4.4.2. Анализ результатов

Качество обучения

На рис. 4.7 представлены результаты сравнительного анализа функций потерь. Рисунок 4.7а демонстрирует долю экспериментов (пар $\langle$*Нейросетевая модель*, *Набор данных*$\rangle$), для которых каждая функция потерь обеспечила наименьшее значение ошибки. Значения приведены в процентах от общего числа рассмотренных экспериментов и отражают частоту достижения наилучшего результата среди всех функций потерь.



(a) Доля экспериментов с лучшей точностью

(b) Средний прирост точности

(c) Среднее падение точности

Рис. 4.7. Сравнение влияния функций потерь на точность нейросетевых моделей восстановления

Рисунок 4.7b иллюстрирует средний относительный прирост точности G_{loss}, вычисленный только для тех экспериментов, где данная функция потерь обеспечивала наименьшую ошибку:

$$G_{\text{loss}} = \frac{\overline{e} - e_{\text{loss}}}{\overline{e}} \cdot 100\%, \tag{4.2}$$

где e_{loss} представляет собой значение ошибки функции потерь для эксперимента, $\overline{e}$ — среднее значение ошибок остальных функций потерь для того же эксперимента. Значение G_{loss} демонстрирует, насколько выбранная функция потерь улучшает качество восстановления относительно среднего уровня других функций. Аналогично, для оценки ситуации, когда функция потерь не обеспечивает наименьшую ошибку, введем показатель среднего падения точности D_{loss} (см. рис. 4.7c):

$$D_{\text{loss}} = \frac{e_{\text{loss}} - \overline{e}}{\overline{e}} \cdot 100\%. \tag{4.3}$$

Показатель D_{loss} отражает, насколько в среднем ошибка, полученная с использованием данной функции потерь, превышает минимальную ошибку среди всех функций потерь для каждого эксперимента. 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. В.2 и В.1 соответственно.

Можно видеть, что предложенная функция потерь MPDE позволяет минимум в 69% случаев повысить точность восстановления в среднем на 44% по сравнению с рассмотренными передовыми функциями потерь. В случае, когда MPDE не обеспечивает наименьшую ошибку, ее отставание от наилучших значений остается невысоким (в среднем 13.7%). Отдельно по группам наборов данных MPDE в среднем опережает аналоги по точности на 43% и 50% и является лучшей в 70% и 69% случаев в группах А и Б соответственно. Высокая частота минимальной ошибки и небольшое значение среднего отставания указывает на стабильность и устойчивость функции потерь. Результаты MPDE предсказуемы, не демонстрируют резких колебаний и сохраняют высокую точность восстановления практически для всех типов данных и моделей.

Табл. 4.6. Параметры MPDE, обеспечивающие наименьшую ошибку

№	Название	Нейросетевая 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. 4.6 приведены значения параметров α и β функции потерь MPDE, при которых нейросетевые модели достигли 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 отнести к одной из указанных выше групп соответственно. Автоматизация подбора гиперпараметров α и β может рассматриваться как одно из направлений будущих исследований.

Изменение значений функций потерь во время обучения

В данной серии экспериментов изучалась динамика изменения значений различных функций потерь и соответствующих норм градиента в процессе обучения нейросетевых моделей. Аналогично исследованиям [68, 135], рассмотрим изменение значений функции потерь по мере уменьшения ошибки восстановления. В этих работах для оценки чувствительности функций потерь происходит постепенное приближение прогнозируемого значения к истинному с фиксацией значения функции потерь и градиента на каждом шаге. Однако такой подход ориентирован на отдельные точки и не учитывает структурные взаимосвязи внутри подпоследовательностей временного ряда. В данной серии экспериментов используется схожий метод, отличие которого заключается в том, что рассматриваются целые подпоследовательности временного ряда вместо отдельных точек. Под «отдалением прогнозируемого значения от истинной точки» в таком случае будет пониматься переход от исходной подпоследовательности, содержащей полезную информацию, к подпоследовательности, состоящей из случайного шума, в которой информация о динамике исходного ряда отсутствует.

Для проведения данного эксперимента генерировалась одномерная подпоследовательность длиной $m = 100$ на основе синусоиды. Значения синусоиды вычислялись равномерно на интервале $[0, 10]$ и использовались в качестве исходной подпоследовательности:

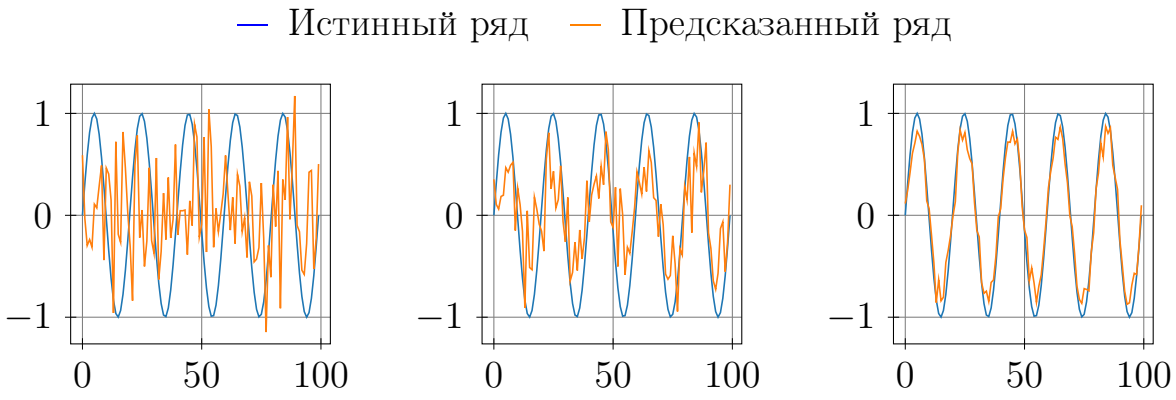
$$Y_{\text{target}} = \{y_i \mid y_i = \sin(x_i), \quad 1 \leqslant i \leqslant m\}, \quad x_i = \frac{10i}{m-1}. \tag{4.4}$$

Для моделирования начального состояния нейросетевой модели формировалась подпоследовательность, состоящая исключительно из случайного шума с нулевым средним и дисперсией 0.25. Такая подпоследовательность Y_{noise} имитирует предсказания нейросетевой модели сразу после инициализации весов случайными значениями, до начала обучения:

$$Y_{\text{noise}} = \{y_i \mid y_i \sim \mathcal{N}(0, 0.25), \quad 1 \leqslant i \leqslant m\}. \tag{4.5}$$

На основе исходной подпоследовательности Y_{target} и ее зашумленной версии Y_{noise} формировался набор $\boldsymbol{Y}_{\text{predict}}$ из $N = 100$ предсказаний нейросетевой модели, имитирующих различные стадии обучения. Каждое предсказание создавалось путем линейного смешивания обеих подпоследовательностей:

$$\begin{gathered} \boldsymbol{Y}_{\text{predict}} = \{Y^{(j)}_{\text{predict}}\}_{j=1}^{N}, \quad Y^{(j)}_{\text{predict}} = \{y_i^{(j)}\}_{i=1}^{m}, \\ y_i^{(j)} = \gamma_j\, Y_{\text{noise}}(i) + (1 - \gamma_j)\, Y_{\text{target}}(i), \;\; \gamma_j = 0.01\, j, \quad 1 \leqslant j \leqslant N. \end{gathered} \tag{4.6}$$

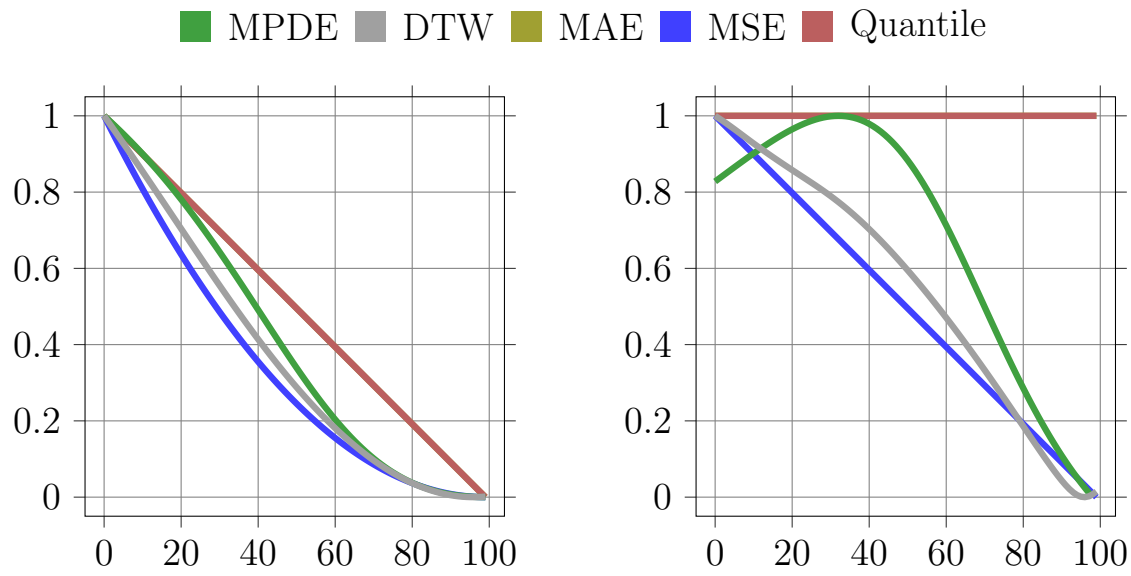


(a) $\gamma = 0.0$ (b) $\gamma = 0.4$ (c) $\gamma = 0.8$

Рис. 4.8. Влияние параметра γ на предсказанные данные

На рис. 4.8 приведен пример сравнения исходной подпоследовательности Y_{target} и предсказаний нейросетевой модели Y_{predict} при различных значениях параметра γ. При больших значениях γ предсказания нейросетевой модели практически совпадают с исходной подпоследовательностью, тогда как при уменьшении γ предсказания содержат больше шума. Значение коэффициента γ плавно увеличиваются с каждого шага, что моделирует по-

степенное приближение предсказаний нейросетевой модели к исходной подпоследовательности на протяжении процесса обучения. Для каждой анализируемой функции потерь на основе набора предсказаний модели формировались два вектора: вектор значений функции потерь и вектор норм градиентов. Норма градиента для каждого предсказания вычислялась как евклидова норма вектора частных производных функции потерь по всем точкам подпоследовательности.



(a) Изменение функции потерь (b) Изменение нормы градиента

Рис. 4.9. Влияние параметра γ на поведение функций потерь

На рис. 4.9 представлены результаты анализа изменения значений функции потерь (см. рис. 4.9a) и норм градиента (см. рис. 4.9b) по мере приближения предсказаний нейросетевой модели к истинной подпоследовательности. Для обеспечения сопоставимости динамики различных функций потерь все графики были нормализованы методом минимаксной нормализации.

Рассмотрим форму кривых каждой функции потерь более подробно. Для функции потерь MAE значения изменяются линейно, отражая прямую зависимость ошибки от отклонения предсказаний. Соответствующая норма градиента остается постоянной, параллельной оси абсцисс, что указывает на одинаковую силу корректирующего воздействия вне зависимости

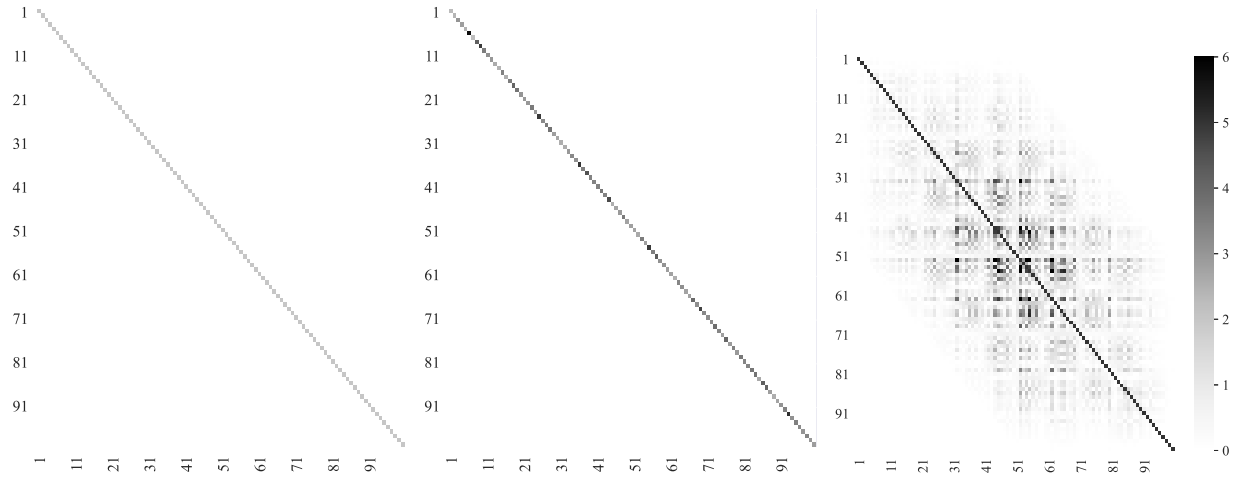
от величины ошибки. Это означает, что MAE не замедляет обучение при больших ошибках, но и не усиливает его, оставаясь равномерной по всей области значений. Из-за стабильности градиента и отсутствия замедления на больших отклонениях, данная функция потерь удобна на ранних этапах обучения и при подборе гиперпараметров. Однако вследствие одинаковой корректировки для малых и больших ошибок, может замедлять достижение высокой точности на последних этапах обучения модели. Аналогичную линейную форму и постоянную норму градиента демонстрирует квантильная функция потерь.

Для функции потерь MSE значения увеличиваются квадратично с ростом ошибки. Норма градиента увеличивается с ростом ошибки, усиливая корректирующее воздействие на нейросетевую модель при больших отклонениях. MSE стимулирует более активное исправление крупных ошибок, что способствует быстрому снижению значительных отклонений на ранних этапах обучения. Однако усиление градиента на больших ошибках может привести к переобучению на выбросах и нестабильности процесса обучения.

График функции потерь DTW имеет нестандартную, сложную форму с чередующимися локальными минимумами и максимумами. Норма градиента изменяется неравномерно на протяжении всего диапазона значений ошибки, отражая неоднородность корректирующих воздействий. Минимум нормы градиента функции потерь достигается не в точках минимального расстояния, а за несколько шагов до них, после чего значение снова увеличивается. Преждевременное снижение скорости изменения функции потерь вблизи локальных экстремумов может приводить к остановке градиента в локальных минимумах до достижения глобального, усложняя оптимизацию.

Для предложенной функции MPDE величина градиента изменяется адаптивно. По мере перехода модели от полностью шумных предсказаний к частично структурированным ($\gamma = 0.2$), норма градиента увеличивается, что отражает усиление корректирующего воздействия на промежуточные ошибки. Далее, по мере приближения предсказаний к истинной подпо-

следовательности, норма градиента постепенно снижается, демонстрируя адаптивное замедление корректировки для уже близких к правильным значениям. Такое поведение позволяет MPDE увеличивать размер обновлений параметров на стадиях обучения, где корректировка наиболее эффективна, снижая риск нестабильного градиентного шага на начальных и финальных этапах.



(a) MSE (b) DTW (c) MPDE

Рис. 4.10. Тепловые карты нормы градиента для различных функций потерь при $\gamma = 0.2$

На рис. 4.10 приведены тепловые карты значений градиента для различных функций потерь при фиксированном значении $\gamma = 0.2$. В данном случае тепловая карта представляет собой матрицу, элементы которой отображают влияние каждой точки подпоследовательности на значения остальных точек при вычислении градиента функции потерь. Каждое значение матрицы с индексами i и j демонстрирует, насколько изменение значения i-й точки влияет на вклад j-й точки в общую функцию потерь.

Строки тепловой карты характеризуют направление и силу корректирующего воздействия. Каждый столбец отражает чувствительность соответствующей точки к изменениям соседних элементов ряда. Цветовая интенсивность пропорциональна норме градиента: более темные участки соответствуют областям с сильным градиентным воздействием, а более светлые соответствуют зонам с низким влиянием.

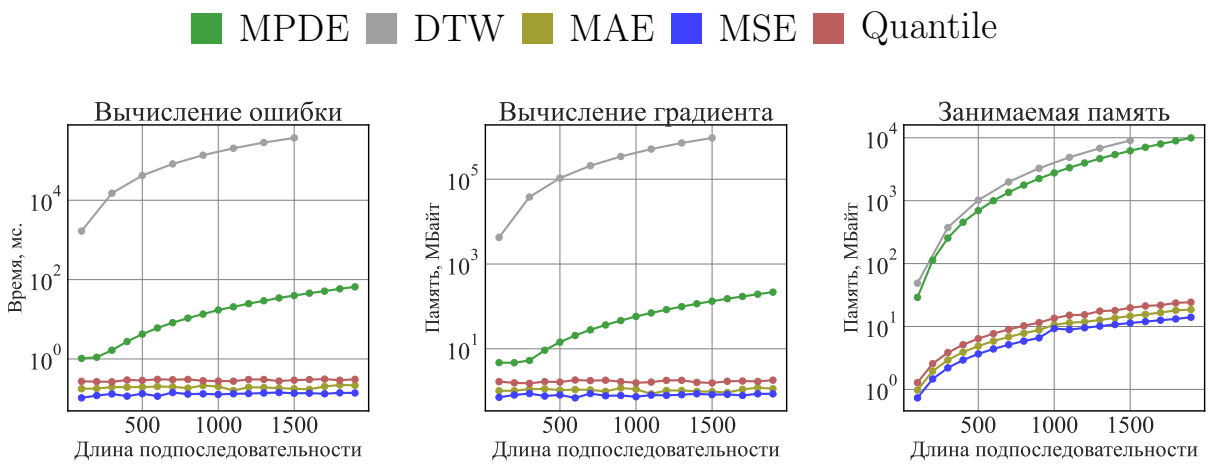
Для MSE (см. рис. 4.10a) каждая точка формирует градиент только для самой себя, при этом величина градиента одинакова для всех точек, что отражает локальный и равномерный характер корректировки. Для DTW (см. рис. 4.10b) наблюдается схожая ситуация: каждая точка вносит вклад только в собственный градиент, однако величины влияния различаются. В случае MPDE (см. рис. 4.10c) заметно более сложное влияние: каждая точка формирует градиент не только для себя, но и для соседних точек в пределах заданного окна. При этом величина градиента на соседние точки меньше, чем у самой точки. В результате MPDE демонстрирует способность учитывать локальные взаимосвязи между точками, что положительно влияет на качество восстановления.

Производительность функций потерь

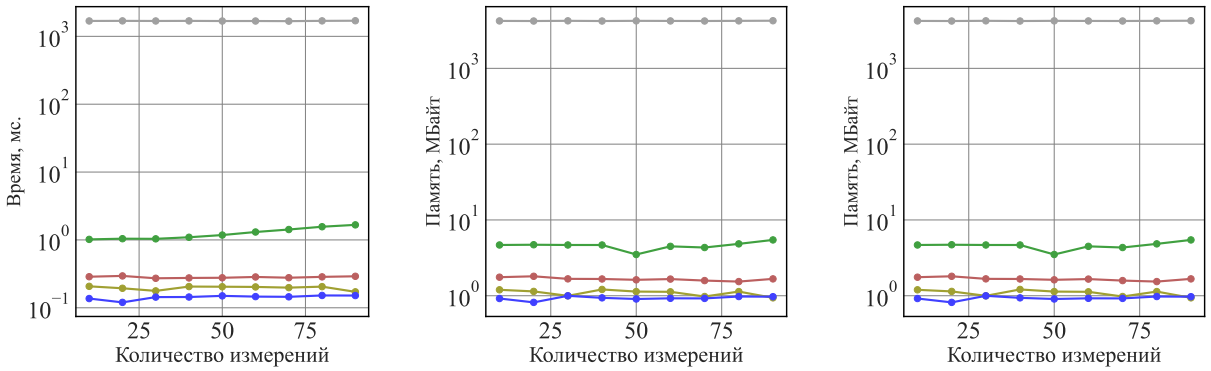
В данной серии экспериментов исследовалась вычислительная эффективность функций потерь. Эксперимент включал три серии, в которых варьировались ключевые параметры входных подпоследовательностей: длина, размер батча и количество измерений. В качестве исходных данных использовались одномерные подпоследовательности на основе синусоиды с добавленным случайным шумом, имитирующим предсказания модели на различных этапах обучения (см. формулу 4.6). В ходе экспериментов измерялись следующие показатели: время вычисления значения функции потерь, время вычисления ее градиента и пиковое использование видеопамяти.

На рис. 4.11 представлены результаты экспериментов. Рисунок состоит из трех групп графиков, каждая из которых показывает зависимость указанных показателей от длины подпоследовательности, количества измерений и размера батча. Временные показатели указаны в миллисекундах, потребление памяти в мегабайтах.

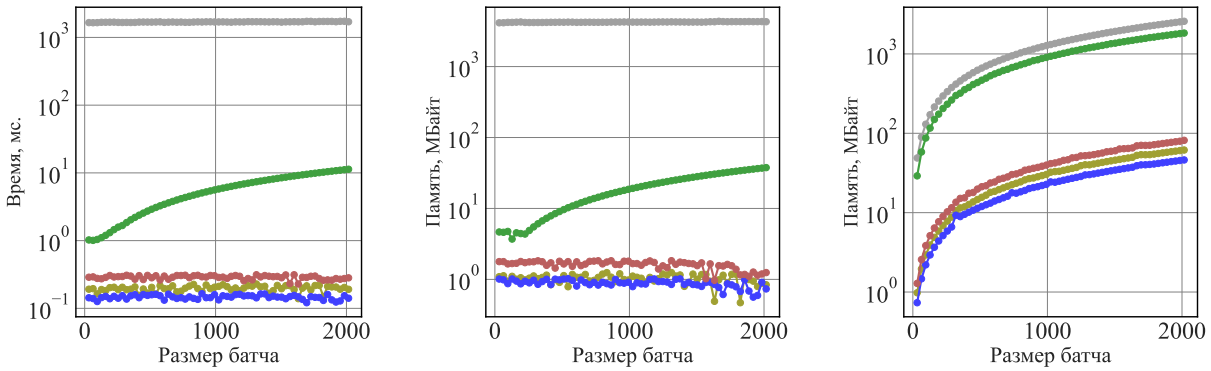
Рис. 4.11a демонстрирует влияние длины подпоследовательности на производительность функций потерь. В этом эксперименте подпоследовательности имели следующие зафиксированные характеристики: количество измерений было равно 10, размер батча 32. С увеличением длины подпоследо-



(a) Влияние длины подпоследовательности



(b) Влияние количества измерений



(c) Влияние размера батча

Рис. 4.11. Сравнение производительности функций потерь

вательности наблюдается существенное различие в вычислительных затратах различных функций потерь. Функция DTW демонстрирует наибольшую вычислительную сложность: время ее выполнения растет почти экспоненциально и на несколько порядков превышает показатели остальных

методов. Функция MPDE также характеризуется ростом вычислительных затрат с увеличением длины подпоследовательности, однако рост происходит медленнее, чем у DTW. Однако MPDE остается более ресурсоемкой по сравнению с классическими функциями потерь (MSE, MAE, Quantile), для которых время вычисления практически не зависит от длины подпоследовательности.

Рис. 4.11b иллюстрирует влияние количества измерений на производительность функций потерь. При фиксированной длине подпоследовательности (100) и размере батча (32) увеличение числа измерений оказывает минимальное влияние на вычислительные характеристики большинства функций потерь. DTW демонстрирует наибольшее время обработки данных. MPDE характеризуется умеренным ростом вычислительных затрат с увеличением размерности, оставаясь значительно быстрее DTW. Классические функции потерь (MSE, MAE, Quantile) демонстрируют стабильные и минимальные значения времени вычисления и потребления памяти.

Рис. 4.11c показывает зависимость производительности функций потерь от размера батча. При фиксированной длине подпоследовательности (100) и числе измерений (10) увеличение размера батча приводит к умеренному росту времени вычислений и потребления видеопамяти для большинства функций потерь. Функция DTW остается наиболее ресурсоемкой. MPDE демонстрирует более низкое время выполнения относительно DTW, однако ее вычислительные затраты растут быстрее, чем у классических функций потерь (MSE, MAE, Quantile).

Результаты экспериментов показывают, что функция MPDE обеспечивает более высокую точность восстановления временных рядов в большинстве случаев. DTW занимает второе место по числу случаев, демонстрирующих высокую точность, однако требует значительно больше вычислительных ресурсов. Классические функции потерь (MSE, MAE, Quantile) характеризуются минимальными вычислительными затратами и демонстрируют хорошую масштабируемость.

Итоговые рекомендации по применению

На основе проведенных вычислительных экспериментов можно сформулировать следующие рекомендации по использованию различных функций потерь. Специализированные функции потерь, такие как MPDE и DTW, могут быть успешно использованы для повышения точности существующих нейросетевых методов восстановления потоковых данных, представленных в форме временных рядов. В случае проектирования и подбора архитектуры нейросетевой модели целесообразно использовать классические функции потерь (MSE, MAE, Quantile), которые обеспечивают высокую скорость вычислений, стабильную сходимость и низкое потребление ресурсов. Использование классических функций потерь на этапе проектирования позволит проводить большее количество итераций обучения, тестировать различные архитектуры, подбирать гиперпараметры и получать первичную оценку качества модели при минимальных вычислительных затратах. Специализированные функции потерь, такие как MPDE и DTW, будут особенно полезны на финальных этапах жизненного цикла в качестве инструмента тонкой настройки нейросетевой модели восстановления потоковых данных, представленных в форме временных рядов.

4.5. Метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления tsGAP

Табл. 4.7. Оборудование, использованное в экспериментах

Характеристики	CPU	GPU (V100)	GPU (RTX 3060)
Бренд и серия	Intel Xeon	NVIDIA Volta	NVIDIA Ampere
Модель	E5-2687W v2	V100	RTX 3060
Количество ядер	8	5 120	3 584
Тактовая частота, ГГц	3,40	1,53	1,78
Память, ГБ	16	32	12

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты, в которых использовалось оборудование Лаборатории суперкомпьютерного моделирования ЮУрГУ [1]. В

табл. 4.7 приведены характеристики оборудования, задействованного при исследовании пространства поиска и обучения моделей предсказания качества нейросетевых моделей. Вычислительные эксперименты, связанные с исследованием пространства поиска, осуществлялись в течение трех месяцев.

4.5.1. Описание экспериментов

Область поиска

В рамках данного исследования целевой нейросетевой моделью, исследуемой в ходе нейросетевого поиска, выступает нейросетевая модель, аппроксимирующая функцию восстановления временного ряда $\boldsymbol{T}$.

Формальное определение целевой нейросетевой модели

Для каждой неполной подпоследовательности из множества $\overset{\circ}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$ может быть получена восстановленная подпоследовательность $\overset{\bullet}{\boldsymbol{T}}_{i,\,m}$, в которой значения на местах пропусков заменены восстановленными значениями.

В таком случае задачу восстановления можно формализовать как функцию f_{impute}, устанавливающую соответствие между элементами множества подпоследовательностей с пропущенными значениями $\overset{\circ}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$ и соответствующими им элементами множества восстановленных последовательностей $\overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$. Формально функция восстановления может быть определена следующим образом:

$$f_{\text{impute}} : \overset{\circ}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}} \to \overset{\bullet}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}, \quad f_{\text{impute}}(\overset{\circ}{\boldsymbol{T}}_{i,\,m}) = \overset{\bullet}{\boldsymbol{T}}_{i,\,m}. \tag{4.7}$$

Введем функцию $f_{\text{mask}} \colon \boldsymbol{S}^{m}_{\boldsymbol{T}} \to \overset{\circ}{\boldsymbol{S}}{}^{m}_{\boldsymbol{T}}$, которую можно рассматривать как близкую к обратной функции восстановления. Функция f_{mask} отображает полные подпоследовательности во множество неполных, моделируя появление пропусков. При этом ожидается, что композиция функций f_{impute} и f_{mask} совпадает с тождественным отображением на множестве полных

подпоследовательностей:

$$\overset{\circ}{\boldsymbol{T}}_{i,\,m} = f_{\text{mask}}(\boldsymbol{T}_{i,m}), \quad f_{\text{impute}}(f_{\text{mask}}(\boldsymbol{T}_{i,m})) = \boldsymbol{T}_{i,m}, \quad \boldsymbol{T}_{i,m} \in \boldsymbol{S}^{m}_{\boldsymbol{T}}. \tag{4.8}$$

В соответствии с изложенным выше, целевая нейросетевая модель аппроксимирует функцию f_{impute} и принимает на вход многомерную подпоследовательность $\overset{\circ}{\boldsymbol{T}}_{i,\,m}$, содержащую пропущенные значения, и на выходе формирует восстановленную подпоследовательность $\overset{\bullet}{\boldsymbol{T}}_{i,\,m}$, в которой отсутствующие элементы заменены синтетическими значениями. В процессе обучения минимизируется расхождение между восстановленной подпоследовательностью и соответствующей полной исходной подпоследовательностью.

Обучение целевой модели проводится на обучающей выборке, сформированной на основе временного ряда $\boldsymbol{T}$. Каждое измерение ряда подвергается минимаксной нормализации (см. формулу 2.10) и разбивается на множество подпоследовательностей длины m. На основе полученных подпоследовательностей формируется обучающая выборка как набор кортежей, включающих входные и выходные значения. В процессе обучения в качестве входных данных использовались подпоследовательности, в которые случайным образом добавлялись пропуски. Пропуски вводились до тех пор, пока их доля не достигала 25 % от общего числа элементов в подпоследовательности. Позиции пропусков выбирались равновероятно среди всех элементов соответствующей подпоследовательности. Выходными данными полагаются подпоследовательности до замены на пропуски. Формально обучающая выборка может быть определена следующим образом:

$$\begin{gathered} D = \{\langle \boldsymbol{X},\, \boldsymbol{Y}\rangle \;\mid\; \boldsymbol{X} = \overset{\circ}{\boldsymbol{T}}_{i,\,m}, \quad \boldsymbol{Y} = \boldsymbol{T}_{i,m}, \\ |\{\overset{\circ}{t}{}^{(k)}_{j} \mid \overset{\circ}{t}{}^{(k)}_{j} = \texttt{NAN}\}| = \lfloor 0.25 \cdot m \cdot d \rfloor, \\ \overset{\circ}{\boldsymbol{T}}_{i,\,m} = f_{\text{mask}}(\boldsymbol{T}_{i,m})\}. \end{gathered} \tag{4.9}$$

Ошибка модели рассчитывается как среднеквадратичная ошибка (Mean Squared Error, MSE) между полными подпоследовательностями и восста-

новленными:

$$E_{\text{model}} = \frac{1}{m \cdot d} \sum_{k=1}^{d} \sum_{j=1}^{m} (t_j^{(k)} - \overset{\bullet}{t}_j^{(k)})^2, \quad i \leqslant j \leqslant i+m, \; t_j^{(k)} \in \boldsymbol{T}_{i,m}, \; \overset{\circ}{t}_j^{(k)} \in \overset{\circ}{\boldsymbol{T}}_{i,\,m}.$$

Во время обучения подпоследовательности временного ряда $\boldsymbol{T}$ разделяются на обучающую и тестовую выборки в пропорции 3 к 1. Для повышения устойчивости и объективности оценки качества модели применяется процедура кросс-валидации, состоящая из четырех независимых запусков с различными разбиениями на обучающие и тестовые выборки. Качество модели с точки зрения точности восстановления оценивается на тестовых выборках по пропускам, добавленным искусственно:

$$E_{\text{val}} = \frac{1}{c} \sum_{k=0}^{d} \sum_{j=i}^{i+m} 1_{\{\texttt{NAN}\}}(\overset{\circ}{t}_j^{(k)}) \cdot (t_j^{(k)} - \overset{\bullet}{t}_j^{(k)})^2,$$

$$c = |\{\overset{\circ}{t}_j^{(k)} \mid \overset{\circ}{t}_j^{(k)} = \texttt{NAN}\}|, \; 1_{\{\texttt{NAN}\}}(\overset{\circ}{t}_j^{(k)}) = \begin{cases} 1, \overset{\circ}{t}_j^{(k)} = \texttt{NAN}, \\ 0, \overset{\circ}{t}_j^{(k)} \neq \texttt{NAN}, \end{cases}$$

где $1_{\{\texttt{NAN}\}}(\overset{\circ}{t}_j^{(k)})$ представляет собой индикаторную функцию, принимающую значение 1 в случае, если соответствующее значение искусственно было помечено как пропуск, и 0 в ином случае. Итоговые значения метрик усредняются по всем запускам кросс-валидации.

Параметры целевой нейросетевой модели

В данном исследовании пространство поиска целевой нейросетевой модели было ограничено с учетом характерных особенностей задач восстановления временных рядов. В частности, в качестве базовых компонентов рассматривались типы слоев, обладающие способностью моделировать временные зависимости и широко применяемые в ранее опубликованных работах по анализу и восстановлению временных рядов. Использовались следующие типы слоев: полносвязные (Dense), одномерные сверточные (Conv1D) и рекуррентные (RNN, LSTM, GRU). Для каждого слоя варьировались ин-

дивидуальные параметры (см. табл. 4.8). В качестве базовых компонент нейросетевых моделей рассматривались типы слоев, указанные в таблице С.3, а также функции активации, приведенные в таблице С.1.

Табл. 4.8. Параметры слоев из пространства поиска метода tsGAP

№	Тип слоя	Глубина	Параметры слоя	
			Первый	Второй
1.	Dense	[1,20]	$\{16, 32, 64, 128, 256, 512, 1024\}$	—
2.	Conv1D	[1,5]	$\{32, 64, 128, 256, 512\}$	$\{3, 5, 7\}$
3.	RNN	[1,2]	$\{16, 32, 64, 128\}$	—
4.	LSTM	[1,2]	$\{16, 32, 64, 128\}$	—
5.	GRU	[1,2]	$\{16, 32, 64, 128\}$	—

Для всех рассматриваемых архитектур осуществлялся перебор следующих общих гиперпараметров: длина подпоследовательности принимала значения из множества $\{100, 200, 300\}$ и скорость обучения принимала значения из множества $\{0.001, 0.005, 0.0001, 0.01\}$. Для каждого типа слоя была задана максимально допустимая глубина, определяемая с учетом потенциального риска исчезновения градиента при обучении глубоких нейросетей. Кроме того, для каждого слоя варьировались индивидуальные гиперпараметры, специфичные для соответствующего архитектурного типа. Полный список перебираемых параметров слоев представлен в табл. 4.8.

В совокупности было сформировано дискретное пространство из 200 уникальных моделей, в рамках которого было выполнено более 12 000 запусков с различными параметрами обучения.

Наборы данных, конкуренты и методика сравнения

В качестве наборов данных для экспериментов используются результаты обучения нейросетевых моделей на временных рядах из различных предметных областей. Описание используемых временных рядов представлено в таблице 4.1. В процессе обучения как предлагаемого метода, так и методов-конкурентов, исходный набор данных подвергался разбиению. Из всего множества возможных моделей исключались 25%, оставшиеся 75% использовались для обучения. Исключенные модели использовались для тестирования. Для обеспечения сопоставимости результатов разбиение данных для каждого тестируемого метода оставалось одинаковым.

Во время вычислительных экспериментов сравнивалась ошибка прогноза параметров качества каждого исследуемого метода на тестовой выборке с помощью симметричной средней абсолютной процентной ошибке (Symmetric Mean Absolute Percentage Error, SMAPE). Формально данная ошибка может быть представлена следующим образом:

$$\text{SMAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)\,/2}, \tag{4.10}$$

где y_i — истинное значение, $\hat{y}_i$ — предсказанное нейросетевой моделью значение, n — общее число наблюдений.

В качестве конкурентов использовались следующие методы: XGBoost, NGBoost, LightGBM, Random Forest [162], BOHAMIANN [119], DNGO [117], MLP [142], OMNI [143], VSGP [128]. В качестве реализации сравниваемых методов использовалась реализация, предоставляемая в составе фреймворка NASLib [143]. Гиперпараметры конкурентов подбирались индивидуально для каждого временного ряда с использованием платформы Weights & Biases (wandb) [139] в течение одной недели.

4.5.2. Анализ результатов

На рис. 4.12 представлены результаты вычислительных экспериментов в виде столбчатых диаграмм, отражающих точность прогноза для всех исследуемых методов. Диаграммы организованы в виде таблицы: столбцы соответствуют наборам временных рядов, строки ошибкам прогнозируемых параметров качества. Под параметрами качества подразумеваются ошибка прогноза точности целевой модели и ошибка прогноза времени ее обучения. Каждая диаграмма отображает значения метрики SMAPE для всех сравниваемых методов. Для наглядности наилучшее значение в каждой диаграмме выделено жирным шрифтом.

Метод tsGAP демонстрирует стабильное и значительное преимущество над конкурентами. В среднем по различным наборам данных tsGAP обеспечивает наибольшую точность прогноза как по ошибке модели, так и по

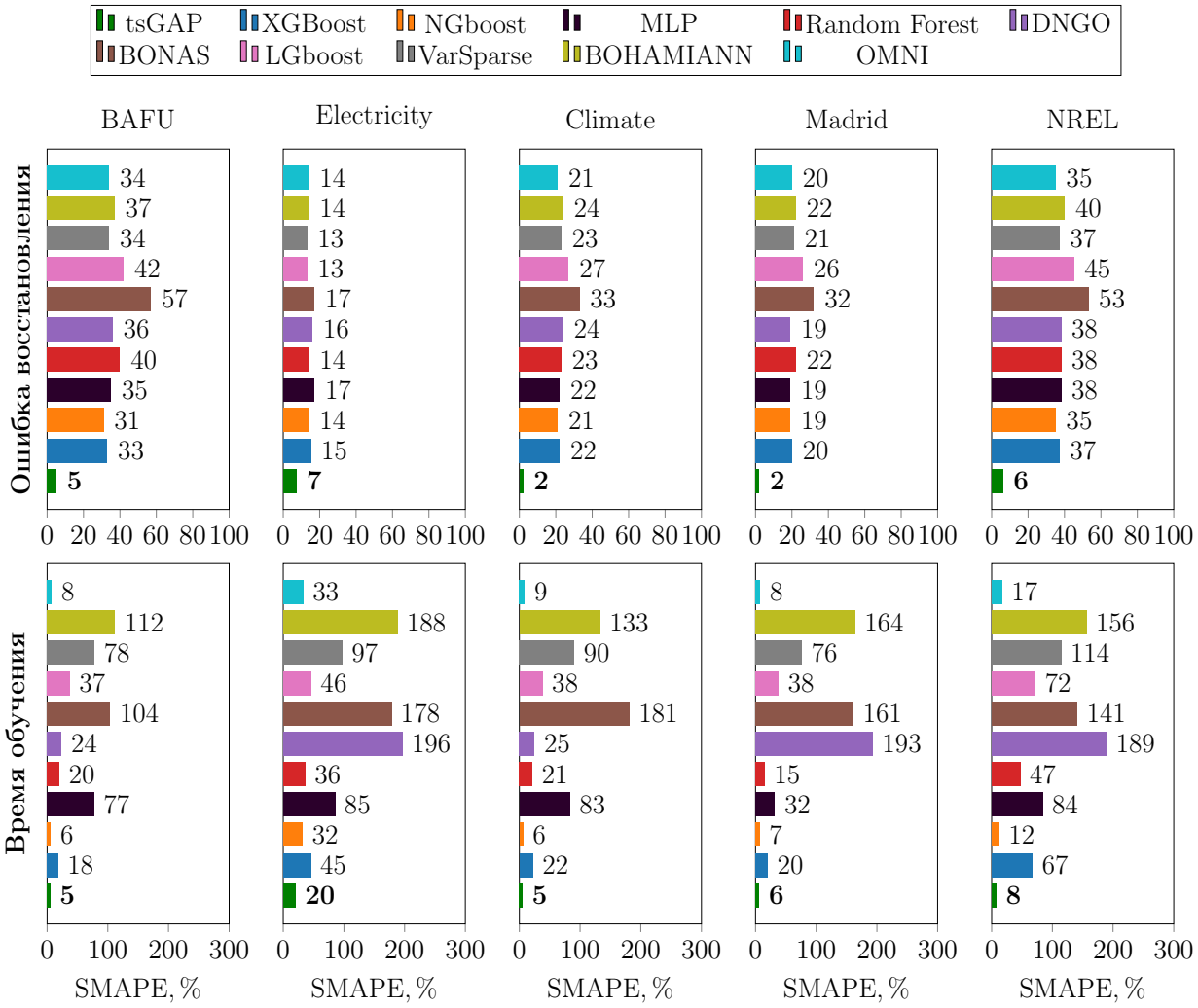


Рис. 4.12. Сравнение ошибки прогнозирования, SMAPE

времени обучения модели. Величина ошибки прогноза, достигаемая предложенным методом, в среднем составляет 4.4 % по ошибке целевой модели и 8.8 % по времени ее обучения. В то же время средние значения аналогичных ошибок среди всех альтернативных подходов составляют 27.6 % и 61.1 % соответственно. Предложенный метод превосходит лучшего конкурента в среднем на 19.6% по прогнозированию ошибки и 4.4% по прогнозированию времени.

Результаты вычислительных экспериментов демонстрируют, что ранжирование исследуемых моделей по точности предсказания ошибки целевой модели не зависит от анализируемого временного ряда.

Данное поведение можно объяснить тем, что рассматриваемые модели не используют временной ряд и его характеристики в качестве входных данных. Соответственно, методы не обладают возможностью учитывать индивидуальные особенности рядов, оказывающие влияние на эффективность архитектуры в задаче восстановления. В результате, прогноз характеристик формируется исключительно на основе информации об архитектуре нейросетевой модели и не зависит от ряда.

Прогноз ошибки целевой модели, как правило, демонстрирует более высокую точность, чем прогноз времени обучения, поскольку точность во многом определяется характеристиками архитектурой модели. Время обучения зависит не только от архитектуры, но и от множества внешних факторов, включая конфигурацию аппаратного обеспечения, специфику реализации операторов в используемых библиотеках, особенности компиляции, выбор программного фреймворка. Указанные параметры, как правило, не представлены явно в доступных данных и обладают высокой степенью вариативности.

4.6. Выводы по главе 4

В данном разделе представлены результаты вычислительных экспериментов, направленных на оценку эффективности предложенных нейросетевых методов и функции потерь. Проведенные исследования включают сравнительный анализ методов SANNI и SAETI с передовыми аналогами из области восстановления временных рядов, исследование функции потерь MPDE в сравнении с традиционными функциями потерь, а также оценку метода tsGAP в сравнении с современными методами предварительной оценки нейросетевых моделей. Эксперименты проводились на реальных и синтетических наборах данных из различных предметных областей. Результаты вычислительных экспериментов демонстрируют следующее.

Метод SANNI демонстрирует наивысшую среднюю точность восстановления в режиме онлайн для временных рядов, содержащих данные с разнообразными активностями субъектов. Для подобных рядов точность ме-

тода превышает средний уровень передовых аналогов на 56%. По сравнению с наилучшим конкурентом достигается преимущество в 17%. Метод SANNI демонстрирует наибольшую точность относительно конкурентов в сценарии Blackout, приближенном к условиям онлайн-режима, успешно восстанавливая пропущенные данные на основе исключительно исторических значений текущего измерения. Проведенный анализ вычислительной эффективности подтверждает, что производительности метода достаточно для его использования в качестве инструментального средства онлайн-восстановления в большинстве прикладных задач.

Метод SAETI превосходит передовые методы восстановления потоковых данных на временных рядах, содержащих различные активности. При обработке подобных рядов метод SAETI уверенно превосходит передовых аналогов: средняя точность метода превышает уровень конкурентов на 63%, а преимущество относительно наилучшего конкурента составляет 22%. Благодаря своей архитектуре метод SAETI может эффективно применяться в качестве инструмента офлайн-восстановления временных рядов.

Нейросетевые модели, обученные с использованием функции потерь MPDE, демонстрируют более высокую точность восстановления по сравнению с моделями, для обучения которых применялись традиционные функции потерь. В частности, в 69% случаев MPDE показала повышение точности в среднем на 44% по сравнению с другими функциями потерь, которые применяются для задачи восстановления временных рядов. Эксперименты по оценке производительности функций потерь показали, что MPDE требует меньших вычислительных ресурсов, чем специализированные методы для временных рядов (например, DTW). Разработанный параллельный алгоритм обеспечивает приемлемую скорость обучения с использованием MPDE, сопоставимую с классическими функциями, позволяющую эффективно интегрировать MPDE в жизненный цикл нейросетевых моделей в качестве функции потерь.

Для проведения вычислительных экспериментов с методами предварительной оценки было сформировано пространство поиска, включающее 200 уникальных архитектур нейросетевых моделей восстановления, обученных

на наборах данных из различных предметных областей. В рамках эксперимента было выполнено более 12 000 запусков обучения целевой нейросетевой модели с различными комбинациями гиперпараметров. Эксперименты на реальных и синтетических данных продемонстрировали высокую точность метода tsGAP: в среднем он превосходит передовые аналоги на 21% по прогнозу ошибки целевой модели и на 56.7% по прогнозу времени обучения. Предложенный метод превосходит лучшего конкурента в среднем на 19.6% по прогнозированию ошибки и 4.4% по прогнозированию времени. Исходя из результатов вычислительных экспериментов, можно заключить, что метод tsGAP может применяется в качестве инструмента предварительной оценки потенциальных нейросетевых архитектур на этапе выбора модели в жизненном цикле нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов.

Таким образом, проведенные вычислительные эксперименты показали, что предложенные методы обеспечивают высокую точность восстановления временных рядов и эффективности анализа нейросетевых моделей, превосходя существующие инструментальные средства реализации различных этапов жизненного цикла нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов.

Результаты, приведенные в этой главе, опубликованы в статьях [8–11, 161].

Заключение

В заключение излагаются итоги диссертационной работы и ее основные отличия от предыдущих работ, даются рекомендации по использованию полученных результатов, а также рассмотрены направления дальнейших исследований в данной области.

Итоги исследования

Данная диссертационная работа посвящена разработке новых и повышению точности существующих моделей, методов и алгоритмов восстановления пропущенных значений в потоковых данных, представленных в форме временных рядов. Актуальность рассматриваемой задачи обусловлена зависимостью эффективности и устойчивости методов интеллектуального анализа, в которых потоковые данные используются в качестве входных данных. В настоящее время обеспечение точного восстановления пропусков потоковых данных является актуальной проблемой, решение которой востребовано в широком спектре научных и практических приложений.

Основным результатом исследования является комплекс нейросетевых моделей, методов и алгоритмов, предназначенный для использования в рамках жизненного цикла нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, и включающий следующие разработки.

1. Метод восстановления потоковых данных, представленных в форме временных рядов, в режиме онлайн SANNI (Snippet and Artificial Neural Network-based Imputation), основанный на совместном использовании поведенческих шаблонов и рекуррентных нейросетей.

2. Метод восстановления потоковых данных, представленных в форме временных рядов, в режиме офлайн SAETI (Snippet based Autoencoder for Timeseries Imputation), основанный на совместном использовании поведенческих шаблонов и автоэнкодеров.

3. Функция потерь MPDE (Mean Profile Distance Error), предназначенная для обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, и оценивающая поведенческую схожесть подпоследовательностей.

4. Метод tsGAP (time-series Graph-Attention Perfomance Prediction model), обеспечивающий прогнозирование ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов, за счет использования представления нейросетевой модели восстановления в виде ориентированного ациклического графа.

Проведены вычислительные эксперименты на синтетических и реальных временных рядах, подтверждающие высокую точность и эффективность предложенных методов и функций в задачах восстановления временных рядов.

Применение полученных результатов

Результаты исследования могут быть использованы в качестве инструментальных средств разработки различных этапов жизненного цикла нейросетевых моделей для повышения качества восстановления потоковых данных, представленных в форме временных рядов и содержащих активности. Разработанные методы SANNI и SAETI обеспечивают восстановление потоковых данных в режимах онлайн и офлайн с высокой точностью. Функция потерь MPDE обеспечивает повышение точности восстановления потоковых данных, представленных в форме временных рядов, выполняемого нейросетевыми моделями. Метод tsGAP позволяет прогнозировать ошибки и оценивать время обучения нейросетевых моделей восстановления без необходимости полного обучения нейросетевой модели. Совокупное применение предложенных методов в интеллектуальных системах, а также их интеграция в платформы AutoML и MLOps, обеспечит автоматизацию обработки пропусков потоковых данных и повысит устойчивость и точность последующего анализа.

Отличия исследования от предыдущих работ

Основные результаты, достигнутые в ходе настоящего диссертационного исследования, представляют собой оригинальный вклад и не повторяют научные разработки, ранее опубликованные другими авторами, обзор которых был представлен в главе 1. Основные отличия заключаются в следующем.

1. Предложены нейросетевые методы SANNI и SAETI для восстановления потоковых данных, представленных в форме временных рядов, в режимах онлайн и офлайн соответственно. В отличие от передовых аналогов, в предложенных методах для повышения точности восстановления используются поведенческие шаблоны, отражающие типичное поведение исследуемого объекта и применяемые в качестве эталонных примеров ожидаемого поведения. Предложенные методы опережают известные аналоги по точности восстановления на данных, описывающих поведение объектов с активностями, в среднем на 17% и 22% для режимов онлайн и офлайн соответственно.

2. Разработана функция потерь MPDE для обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. В отличие от аналогов, MPDE учитывает поведенческое сходство между подпоследовательностями. Результаты вычислительных экспериментов демонстрируют, что использование MPDE обеспечивает повышение точности восстановления в среднем на 44% в 69% случаев по сравнению с передовыми аналогами. Предложен эффективный параллельный алгоритм вычисления MPDE и разработана его реализация для интеграции в современные фреймворки глубокого обучения.

3. Предложен нейросетевой метод tsGAP для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления потоковых данных, представленных в форме временных рядов. В отличие от существующих подходов к прогнозированию качества целевых нейросете-

вых моделей восстановления, для повышения точности прогнозов в методе tsGAP используются графовые нейронные сети, анализирующие нейросетевую модель восстановления, представленную в виде ориентированного ациклического графа. Вычислительные эксперименты, проведенные на реальных и синтетических данных, продемонстрировали превосходство метода tsGAP над передовыми аналогами в среднем на 38.8%. Предложенный метод превосходит лучшего конкурента в среднем на 19.6% по прогнозированию ошибки и 4.4% по прогнозированию времени обучения.

Исходные тексты программ, реализующие разработанные в рамках диссертационного исследования методы SANNI, SAETI и tsGAP, функцию потерь MPDE, вместе с наборами данных, которые использовались в вычислительных экспериментах, размещены в свободном доступе в репозиториях сети Интернет [157–160].

Направления будущих исследований

Теоретические исследования и практические разработки, выполненные в рамках этой диссертационной работы, предполагается продолжить по следующим направлениям:

— разработка методов и моделей восстановления потоковых данных, представленных в форме временных рядов, на основе поведенческих шаблонов, позволяющих производить трансферное обучение;

— разработка алгоритма предварительной оценки пропусков потоковых данных, представленных в форме временных рядов, на основе использования поведенческих шаблонов;

— повышение эффективности вычисления функции потерь MPDE;

— разработка средств автоматизации проектирования нейросетевые модели восстановления потоковых данных, представленных в форме временных рядов, на основе метода tsGAP.

Финансовая поддержка исследования

Диссертационное исследование выполнено при финансовой поддержке стипендии Президента Российской Федерации № SPN.2025.04099 (2025 г.) и Российского научного фонда (грант № 23-21-00465).

Список литературы

1. 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.
2. Воеводин В.В., Стефанов К.С. Создание переносимого программного комплекса для мониторинга и анализа производительности суперкомпьютерных приложений // Вычислительные методы и программирование. 2023. Т. 24. С. 24–36. DOI: 10.26089/NumMet.v24r103.
3. Громов В.А., Лукьянченко П.П., Бесчастнов Ю.Н., Томащук К.К. Анализ структуры временных рядов количества дел в суде // Вестник кибернетики. 2022. 4 (48). С. 37–48. DOI: 10.34822/1999-7604-2022-4-37-48.
4. Дышаев М.М., Соколинская И.М. Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2013. Т. 2, № 4. С. 103–108.
5. Иванов С.А., Никольская К.Ю., Радченко Г.И. и др. Концепция построения цифрового двойника города // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 5–23. DOI: 10.14529/cmse200401.
6. Краева Я.А. Нейросетевой метод обнаружения аномалий в многомерных потоковых временных рядах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 4. С. 35–52. DOI: 10.14529/cmse240403.
7. Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник Южно-Уральского государственного университе-

та. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 16–36. DOI: 10.14529/cmse210302.

8. 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. Т. 24, № 3. С. 243–259. DOI: 10.26089/NumMet.v24r318.

9. Юртин А.А. Восстановление многомерных временных рядов на основе выявления поведенческих шаблонов и применения автоэнкодеров // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 39–55. DOI: doi.org/10.14529/cmse240203.

10. 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.

11. Юртип А.А. Метод прогнозирования ошибки времени обучения нейросетевых моделей восстановления многомерных временных рядов // Проблемы информатики. 2025. № 3. С. 72–95. DOI: 10.24412/2073-0667-2025-3-72-95.

12. Adhikari D., Jiang W., Zhan J., *et al.* A Comprehensive Survey on Imputation of Missing Data in Internet of Things // ACM Comput. Surv. 2023. Vol. 55, no. 7. 133:1–133:38. DOI: 10.1145/3533381.

13. Afifi A.A., Elashoff R.M. Missing Observations in Multivariate Statistics: I. Review of the Literature // Journal of the American Statistical Association. 1966. Vol. 61, no. 315. P. 595–604. DOI: 10.2307/2282773.

14. Afrifa-Yamoah E., Mueller U.A., Taylor S.M., Fisher A.J. Missing data imputation of high-resolution temporal climate time series data // Me-

teorological Applications. 2020. Vol. 27, no. 1. e1873. DOI: 10.1002/met.1873.

15. Ahn H., Sun K., Kim K.P. Comparison of Missing Data Imputation Methods in Time Series Forecasting // Computers, Materials & Continua. 2022. Vol. 70, no. 1. P. 767–779. DOI: 10.32604/cmc.2022.019369.

16. Almeida A., Brás S., Sargento S., Pinto F.C. Time series big data: a survey on data stream frameworks, analysis and algorithms // Journal of Big Data. 2023. Vol. 10, no. 1. P. 83. DOI: 10.1186/s40537-023-00760-1.

17. Antoniou C., Dilaveris P., Manolakou P., *et al.* QT Prolongation and Malignant Arrhythmia: How Serious a Problem? // European Cardiology. 2017. Vol. 12, no. 2. P. 112–120. DOI: 10.15420/ecr.2017:16:1.

18. Ashmore R., Calinescu R., Paterson C. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges // ACM Comput. Surv. 2022. Vol. 54, no. 5. 111:1–111:39. DOI: 10.1145/3453444.

19. Aydin S. Time series analysis and some applications in medical research // Journal of Mathematics and Statistics Studies. 2022. Vol. 3, no. 2. P. 31–36. DOI: 10.32996/jmss.2022.3.2.3.

20. Baldi P. Autoencoders, Unsupervised Learning, and Deep Architectures // Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011. Vol. 27 / ed. by I. Guyon, G. Dror, V. Lemaire, *et al.* JMLR.org, 2012. P. 37–50. (JMLR Proceedings). URL: http://proceedings.mlr.press/v27/baldi12a.html.

21. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case // Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

22. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning // Appl. Artif. Intell. 2003. Vol. 17, no. 5/6. P. 519–533. DOI: 10.1080/713827181.

23. 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.

24. Beck N. Estimating Dynamic Models Using Kalman Filtering // Political Analysis. 1989. Vol. 1. P. 121–156. DOI: 10.1093/pan/1.1.121.

25. 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.

26. Bridle J.S. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition // Neurocomputing / ed. by F.F. Soulié, J. Hérault. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990. P. 227–236. DOI: 978-3-642-76153-9_28.

27. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. (Accessed: 3.9.2023). https://www.hydrodaten.admin.ch/.

28. Butera N.M., Li S., Evenson K.R., *et al.* Hot Deck Multiple Imputation for Handling Missing Accelerometer Data // Statistics in Biosciences. 2019. Vol. 11, no. 2. P. 422–448. DOI: 10.1007/s12561-018-9225-4.

29. Cai J., Candès E.J., Shen Z. A Singular Value Thresholding Algorithm for Matrix Completion // SIAM J. Optim. 2010. Vol. 20, no. 4. P. 1956–1982. DOI: 10.1137/080738970.

30. 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.

31. Carvalho Jr. O.A. de, Guimarães R.F., Gomes R.A.T., Silva N.C. da. Time series interpolation // Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS). Barcelona, Spain: IEEE, 07/2007. P. 1959–1961. DOI: 10.1109/I-GARSS.2007.4423211.

32. Ceylan Y., Sipan A., Cem I., Inci B. Comparison of missing value imputation methods in time series: the case of Turkish meteorological data // Theoretical and Applied Climatology. 2013. Vol. 112, no. 1. P. 143–167. DOI: 10.1007/s00704-012-0723-x.

33. 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.

34. Chen Y., Chen Q., Wang R. Bearing Fault Diagnosis Based on Vibration Envelope Spectral Characteristics // Applied Sciences. 2025. Vol. 15, no. 4. DOI: 10.3390/app15042240.

35. Cho K., Merrienboer B. van, Bahdanau D., Bengio Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches // Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014 / ed. by D. Wu, M. Carpuat, X. Carreras, E.M. Vecchi. Association for Computational Linguistics, 2014. P. 103–111. DOI: 10.3115/V1/W14-4012.

36. Chung J., Gülçehre Ç., Cho K., Bengio Y. Gated Feedback Recurrent Neural Networks // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. Vol. 37 / ed. by F.R. Bach, D.M. Blei. JMLR.org, 2015. P. 2067–2075. URL: http://proceedings.mlr.press/v37/chung15.html.

37. Clevert D., Unterthiner T., Hochreiter S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) // 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings / ed. by Y. Bengio, Y. LeCun. 2016.

38. Cuturi M., Blondel M. Soft-DTW: a Differentiable Loss Function for Time-Series // Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Vol. 70 / ed. by D. Precup, Y.W. Teh. PMLR, 2017. P. 894–903. URL: http://proceedings.mlr.press/v70/cuturi17a.html.

39. Cybenko G. Approximation by superpositions of a sigmoidal function // Mathematics of Control, Signals and Systems. 1989. Dec. Vol. 2, no. 4. P. 303–314. DOI: 10.1007/BF02551274.

40. Dempster A.P., Laird N.M., Rubin D.B. Maximum likelihood from incomplete data via the EM algorithm // Journal of the Royal Statistical Society: Series B (Methodological). 1977. Vol. 39, no. 1. P. 1–38. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

41. Deng Y., Han C., Guo J., *et al.* Online Missing Data Imputation Using Virtual Temporal Neighbor in Wireless Sensor Networks // Wireless Communications and Mobile Computing. 2022. Feb. Vol. 2022. P. 4909476. DOI: 10.1155/2022/4909476.

42. Ding Y., Huang Z., Shou X., *et al.* Architecture-Aware Learning Curve Extrapolation via Graph Ordinary Differential Equation // AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA / ed. by T. Walsh, J. Shah, Z. Kolter. AAAI Press, 2025. P. 16289–16297. DOI: 10.1609/AAAI.V39I15.33789.

43. Dong X., Yang Y. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search // 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April

26-30, 2020. 2020. URL: https://openreview.net/forum?id=HJxyZkBKDr.

44. 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.

45. Esling P., Agon C. Time-series data mining // ACM Comput. Surv. New York, NY, USA, 2012. Dec. Vol. 45, no. 1. DOI: 10.1145/2379776.2379788.

46. Esling P., Agón C. Time-series data mining // ACM Comput. Surv. 2012. Vol. 45, no. 1. 12:1–12:34. DOI: 10.1145/2379776.2379788.

47. Fang C., Wang C. Time Series Data Imputation: A Survey on Deep Learning Approaches // CoRR. 2020. Vol. abs/2011.11347. URL: https://arxiv.org/abs/2011.11347.

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

49. 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.

50. Fu T. A review on time series data mining // Engineering Applications of Artificial Intelligence. 2011. Vol. 24, no. 1. P. 164–181. DOI: 10.1016/j.engappai.2010.09.007.

51. Gawlikowski J., Tassi C.R.N., Ali M., *et al.* A survey of uncertainty in deep neural networks // Artificial Intelligence Review. 2023. Vol. 56, no. 1. P. 1513–1589. DOI: 10.1007/s10462-023-10562-9.

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

53. 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.

54. Gill M.K., Asefa T., Kaheil Y., McKee M. Effect of missing data on performance of learning algorithms for hydrologic predictions: Implications to an imputation technique // Water Resources Research. 2007. Vol. 43, no. 7. DOI: 10.1029/2006WR005298.

55. 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.

56. Guo Y., Li S., Lerman G. The effect of Leaky ReLUs on the training and generalization of overparameterized networks // International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain. Vol. 238 / ed. by S. Dasgupta, S. Mandt, Y. Li. PMLR, 2024. P. 4393–4401. URL: https://proceedings.mlr.press/v238/guo24c.html.

57. Haitovsky Y. Missing Data in Regression Analysis // Journal of the Royal Statistical Society. Series B (Methodological). 1968. Vol. 30, no. 1. P. 67–82. DOI: 10.1111/j.2517-6161.1968.tb01507.x.

58. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.

IEEE Computer Society, 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.

59. He X., Zhao K., Chu X. AutoML: A survey of the state-of-the-art // Knowl. Based Syst. 2021. Vol. 212. P. 106622. DOI: 10.1016/J.KNOSYS.2020.106622.

60. Historical Crypto Data. https://www.cryptodatadownload.com/data/.

61. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/S0218488598000094.

62. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Comput. 1997. Vol. 9, no. 8. P. 1735–1780. DOI: 10.1162/NECO.1997.9.8.1735.

63. Hornik K. Approximation capabilities of multilayer feedforward networks // Neural Networks. 1991. Jan. Vol. 4, no. 2. P. 251–257. DOI: 10.1016/0893-6080(91)90009-T.

64. Huber P.J. Robust Estimation of a Location Parameter // Breakthroughs in Statistics: Methodology and Distribution / ed. by S. Kotz, N.L. Johnson. New York, NY: Springer New York, 1992. P. 492–518. DOI: 10.1007/978-1-4612-4380-9_35.

65. Imani S., Keogh E.J. Matrix Profile XIX: Time Series Semantic Motifs: A New Primitive for Finding Higher-Level Structure in Time Series // 2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019 / ed. by J. Wang, K. Shim, X. Wu. IEEE, 2019. P. 329–338. DOI: 10.1109/ICDM.2019.00043.

66. Imani S., Madrid F., Ding W., *et al.* Introducing time series snippets: A new primitive for summarizing long time series // Data Min. Knowl. Discov. 2020. Vol. 34, no. 6. P. 1713–1743. DOI: 10.1007/s10618-020-00702-y.

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

68. 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.

69. Jin X., Yu X., Wang X., *et al.* Prediction for Time Series with CNN and LSTM // Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019) / ed. by R. Wang, Z. Chen, W. Zhang, Q. Zhu. Singapore: Springer Singapore, 2020. P. 631–641. DOI: 10.1007/978-981-15-0474-7_59.

70. Kashpruk N., Piskor-Ignatowicz C., Baranowski J. Time Series Prediction in Industry 4.0: A Comprehensive Review and Prospects for Future Advancements // Applied Sciences. 2023. Vol. 13, no. 22. DOI: 10.3390/app132212374.

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

72. 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.

73. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams // Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.

74. Khayati M., Böhlen M.H., Gamper J. Memory-efficient centroid decomposition for long time series // IEEE 30th International Conference on

Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014 / ed. by I.F. Cruz, E. Ferrari, Y. Tao, *et al.* IEEE Computer Society, 2014. P. 100–111. DOI: 10.1109/ICDE.2014.6816643.

75. 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. Vol. 62, no. 6. P. 2257–2280. DOI: 10.1007/S10115-019-01421-7.

76. 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.

77. Kingma D.P., Welling M. An Introduction to Variational Autoencoders // Foundations and Trends in Machine Learning. 2019. Vol. 12, no. 4. P. 307–392. DOI: 10.1561/2200000056.

78. Kosobud R. A Note on a Problem Caused by Assignment of Missing Data in Sample Surveys // Econometrica. 1963. Vol. 31, no. 3. P. 562–563. DOI: 10.2307/1909998.

79. Kreuzer T., Zdravkovic J., Papapetrou P. Unpacking the trend: decomposition as a catalyst to enhance time series forecasting models // Data Mining and Knowledge Discovery. 2025. Vol. 39, no. 5. P. 54. DOI: 10.1007/s10618-025-01120-8.

80. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review // J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/S40537-019-0268-2.

81. 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.

82. Landauer M., Skopik F., Stojanovic B., *et al.* A review of time-series analysis for cyber security analytics: from intrusion detection to attack prediction // Int. J. Inf. Sec. 2025. Vol. 24, no. 1. P. 3. DOI: 10.1007/S10207-024-00921-0.

83. Lara-Benítez P., Carranza-García 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. DOI: 10.3390/app10072322.

84. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. DOI: 10.17882/39754.

85. Lettenmaier D.P. Detection of trends in water quality data from records with dependent observations // Water Resources Research. 1976. Vol. 12, no. 5. P. 1037–1046. DOI: 10.1029/WR012i005p01037.

86. Li C., Liu K., Liu S. A Survey of Loss Functions in Deep Learning // Mathematics. 2025. Vol. 13, no. 15. DOI: 10.3390/math13152417.

87. Li C., Shen J., Yang Y., *et al.* Repetitive Activity Monitoring from Multivariate Time Series: A Generic and Efficient Approach // IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems, MASS 2021, Denver, CO, USA, October 4-7, 2021. IEEE, 2021. P. 36–45. DOI: 10.1109/MASS52906.2021.00014.

88. Li L., McCann J., Pollard N.S., Faloutsos C. DynaMMo: mining and summarization of coevolving sequences with missing values // 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.E. IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 507–516. DOI: 10.1145/1557019.1557078.

89. Liu C.L., Layland J.W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment // J. ACM. New York, NY, USA, 1973. Jan. Vol. 20, no. 1. P. 46–61. DOI: 10.1145/321738.321743.

90. Liu C., Zhu L., Belkin M. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks // Applied and Computational Harmonic Analysis. 2022. Vol. 59. P. 85–116. Special Issue on Harmonic Analysis and Machine Learning DOI: https://doi.org/10.1016/j.acha.2021.12.009.

91. Liu L., Peng Y., Wang S., *et al.* Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors // Information Sciences. 2016. Vol. 340/341. P. 41–57. DOI: 10.1016/j.ins.2016.01.020.

92. Liu Y., Yu R., Zheng S., *et al.* NAOMI: Non-Autoregressive Multiresolution Sequence Imputation // 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. 11236–11246. URL: https://proceedings.neurips.cc/paper/2019/hash/50c1f44e426560f3f2cdcb3e19e39903-Abstract.html.

93. 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.E. IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

94. Lu J., Liu A., Dong F., *et al.* Learning under Concept Drift: A Review // IEEE Trans. Knowl. Data Eng. 2019. Vol. 31, no. 12. P. 2346–2363. DOI: 10.1109/TKDE.2018.2876857.

95. Lu L., Yeonjong S., Yanhui S., Karniadakis George E. Dying ReLU and Initialization: Theory and Numerical Examples // Communications in Computational Physics. 2020. Vol. 28, no. 5. P. 1671–1706. DOI: 10.4208/cicp.OA-2020-0165.

96. Ma Z., Xiao M., Xiao Y., *et al.* High-Reliability and Low-Latency Wireless Communication for Internet of Things: Challenges, Fundamentals, and Enabling Technologies // IEEE Internet Things J. 2019. Vol. 6, no. 5. P. 7946–7970. DOI: 10.1109/JIOT.2019.2907245.

97. Mazumder R., Hastie T., Tibshirani R. Spectral Regularization Algorithms for Learning Large Incomplete Matrices // J. Mach. Learn. Res. 2010. Vol. 11. P. 2287–2322. DOI: 10.5555/1756006.1859931.

98. Mei J., Castro Y. de, Goude Y., Hébrail G. Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates // Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Vol. 70 / ed. by D. Precup, Y.W. Teh. PMLR, 2017. P. 2382–2390. URL: http://proceedings.mlr.press/v70/mei17a.html.

99. MeteoSwiss: Federal Office of Meteorology and Climatology. 2023. (Accessed: 3.9.2023). https://www.meteoswiss.admin.ch/services-and-publications/service/open-government-data.html.

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

101. Moritz S., Bartz-Beielstein T. imputeTS: Time Series Missing Value Imputation in R // R J. 2017. Vol. 9, no. 1. P. 207. DOI: 10.32614/RJ-2017-009.

102. Mutschler C., Ziekow H., Jerzak Z. The DEBS 2013 grand challenge // The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA, June 29 - July 03, 2013 / ed. by S. Chakravarthy, S.D. Urban, P.R. Pietzuch, E.A. Rundensteiner. ACM, 2013. P. 289–294. DOI: 10.1145/2488222.2488283.

103. Niako N., Melgarejo J.D., Maestre G.E., Vatcheva K.P. Effects of missing data imputation methods on univariate blood pressure time series

data analysis and forecasting with ARIMA and LSTM // BMC Medical Research Methodology. 2024. Vol. 24, no. 1. P. 320. DOI: 10.1186/s12874-024-02448-3.

104. Papadimitriou S., Sun J., Faloutsos C., Yu P.S. Dimensionality Reduction and Filtering on Time Series Sensor Streams // Managing and Mining Sensor Data / ed. by C.C. Aggarwal. Springer, 2013. P. 103–141. DOI: 10.1007/978-1-4614-6309-2_5.

105. 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.

106. Paulhus J.L., Kohler M.A. Interpolation of missing precipitation records // Monthly Weather Review. 1952. Vol. 80, no. 8. P. 129–133.

107. 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.

108. Pratama I., Permanasari A.E., Ardiyanto I., Indrayani R. A review of missing values handling methods on time-series data // 2016 International Conference on Information Technology Systems and Innovation (ICITSI). 2016. P. 1–6. DOI: 10.1109/ICITSI.2016.7858189.

109. 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.

110. Reiss A., Stricker D. Introducing a New Benchmarked Dataset for Activity Monitoring // 16th International Symposium on Wearable Computers, ISWC 2012, Newcastle, United Kingdom, June 18-22, 2012. IEEE Computer Society, 2012. P. 108–109. DOI: 10.1109/ISWC.2012.13.

111. Roberts S.J., Osborne M.A., Ebden M., *et al.* Gaussian processes for time-series modelling // Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2013. Vol. 371, no. 1984. P. 20110550. DOI: 10.1098/rsta.2011.0550.

112. 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.

113. Shaw P., Uszkoreit J., Vaswani A. Self-Attention with Relative Position Representations // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers) / ed. by M.A. Walker, H. Ji, A. Stent. Association for Computational Linguistics, 2018. P. 464–468. DOI: 10.18653/V1/N18-2074.

114. Sheppy M., Beach A., Pless S. NREL RSF Measured Data 2011. 11/2014. Accessed: 2023-09-03 DOI: 10.25984/1845288.

115. Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network // Physica D: Nonlinear Phenomena. 2020. Vol. 404. P. 132306. DOI: https://doi.org/10.1016/j.physd.2019.132306.

116. Shu X., Porikli F., Ahuja N. Robust Orthonormal Subspace Learning: Efficient Recovery of Corrupted Low-Rank Matrices // 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014. IEEE Computer Society, 2014. P. 3874–3881. DOI: 10.1109/CVPR.2014.495.

117. Snoek J., Rippel O., Swersky K., *et al.* Scalable Bayesian Optimization Using Deep Neural Networks // Proceedings of the 32nd International Conference on Machine Learning (ICML). Vol. 37. Lille, France: PMLR, 2015. P. 2171–2180.

118. Souza A.S., Bezerra M.A., Cerqueira U.M.F.M., *et al.* An introductory review on the application of principal component analysis in the data exploration of the chemical analysis of food samples // Food Science and Biotechnology. 2024. Vol. 33, no. 6. P. 1323–1336. DOI: 10.1007/s10068-023-01509-5.

119. Springenberg J.T., Klein A., Falkner S., Hutter F. Bayesian optimization with robust Bayesian neural networks // Proceedings of the 30th International Conference on Neural Information Processing Systems. Barcelona, Spain: Curran Associates Inc., 2016. P. 4141–4149. DOI: 10.5555/3157382.3157560.

120. Srivastava N., Hinton G.E., Krizhevsky A., *et al.* Dropout: a simple way to prevent neural networks from overfitting // J. Mach. Learn. Res. 2014. Vol. 15, no. 1. P. 1929–1958. DOI: 10.5555/2627435.2670313.

121. Strubell E., Ganesh A., McCallum A. Energy and Policy Considerations for Deep Learning in NLP // Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers / ed. by A. Korhonen, D.R. Traum, L. Màrquez. Association for Computational Linguistics, 2019. P. 3645–3650. DOI: 10.18653/V1/P19-1355.

122. Sugawara Y., Shiota S., Kiya H. Checkerboard artifacts free convolutional neural networks // APSIPA Transactions on Signal and Information Processing. 2019. Vol. 8. e9. DOI: 10.1017/ATSIP.2019.2.

123. 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.

124. Tealab A. Time series forecasting using artificial neural networks methodologies: A systematic review // Future Computing and Informatics Journal. 2018. Vol. 3, no. 2. P. 334–340. DOI: 10.1016/j.fcij.2018.10.003.

125. Terven J.R., Córdova-Esparza D., Romero-González J., *et al.* A comprehensive survey of loss functions and metrics in deep learning // Artif. Intell. Rev. 2025. Vol. 58, no. 7. P. 195. DOI: 10.1007/S10462-025-11198-7.

126. Testi M., Ballabio M., Frontoni E., *et al.* MLOps: A Taxonomy and a Methodology // IEEE Access. 2022. Vol. 10. P. 63606–63618. DOI: 10.1109/ACCESS.2022.3181730.

127. Thakur S., Choudhary J., Singh D.P. A Survey on Missing Values Handling Methods for Time Series Data // Intelligent Systems / ed. by A. Sheth, A. Sinhal, A. Shrivastava, A.K. Pandey. Singapore: Springer Singapore, 2021. P. 435–443. DOI: 10.1007/978-981-16-2248-9_42.

128. Titsias M. Variational Learning of Inducing Variables in Sparse Gaussian Processes // Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics. Vol. 5 / ed. by D. van Dyk, M. Welling. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr/2009. P. 567–574. URL: https://proceedings.mlr.press/v5/titsias09a.html.

129. Torres J.F., Hadjout D., Sebaa A., *et al.* Deep Learning for Time Series Forecasting: A Survey // Big Data. 2021. Feb. Vol. 9, no. 1. P. 3–21. Epub 2020 Dec 3 DOI: 10.1089/big.2020.0159.

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

131. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays // Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/BIOINFORMATICS/17.6.520.

132. Vaswani A., Shazeer N., Parmar N., *et al.* Attention is All you Need // Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA / ed. by I. Guyon, U. von Luxburg, S. Bengio, *et al.* 2017. P. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

133. Velickovic P., Cucurull G., Casanova A., *et al.* Graph Attention Networks // 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. 2018. DOI: 10.17863/CAM.48429.

134. Wang C., Chen X., Wu C., Wang H. AutoTS: Automatic Time Series Forecasting Model Design Based on Two-Stage Pruning // CoRR. 2022. Vol. abs/2203.14169. DOI: 10.48550/ARXIV.2203.14169.

135. 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.

136. Wang X., Mueen A., Ding H., *et al.* Experimental comparison of representation methods and distance measures for time series data // Data Min. Knowl. Discov. 2013. Vol. 26, no. 2. P. 275–309. DOI: 10.1007/S10618-012-0250-5.

137. Wang Y., Wu H., Dong J., *et al.* Deep Time Series Models: A Comprehensive Survey and Benchmark // CoRR. 2024. Vol. abs/2407.13278. DOI: 10.48550/ARXIV.2407.13278. arXiv: 2407.13278.

138. Weather Station Saaleaue, Max Planck Institute for Biogeochemistry, Germany. (Accessed: 3.9.2023). https://www.bgc-jena.mpg.de/wetter/weather_data.html.

139. Weights & Biases: Machine learning experiment tracking, dataset versioning, and model management. 2020–2025. URL: https://wandb.ai/ (accessed: 6.6.2025).

140. Wellenzohn K., Böhlen M.H., Dignös A., *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series // Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017 / ed. by V. Markl, S. Orlando, B. Mitschang, *et al.* OpenProceedings.org, 2017. P. 330–341. DOI: 10.5441/002/EDBT.2017.30.

141. Wen Q., Zhou T., Zhang C., *et al.* Transformers in Time Series: A Survey // Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China. ijcai.org, 2023. P. 6778–6786. DOI: 10.24963/IJCAI.2023/759.

142. White C., Neiswanger W., Savani Y. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search // Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. AAAI Press, 2021. P. 10293–10301. DOI: 10.1609/AAAI.V35I12.17233.

143. White C., Zela A., Ru B., *et al.* How powerful are performance predictors in neural architecture search? // Proceedings of the 35th International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc., 2021. P. 28454–28469.

144. Williams R.J., Zipser D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks // Neural Comput. 1989. Vol. 1, no. 2. P. 270–280. DOI: 10.1162/NECO.1989.1.2.270.

145. Wilson S.J. Data representation for time series data mining: time domain approaches // Wiley Interdisciplinary Reviews: Computational Statistics. 2017. Vol. 9, no. 1. e1392. DOI: 10.1002/wics.1392.

146. Wu S.-F., Chang C.-Y., Lee S.-J. Time series forecasting with missing values // 2015 1st International Conference on Industrial Networks

and Intelligent Systems (INISCom). 2015. P. 151–156. DOI: 10.4108/icst.iniscom.2015.258269.

147. Wu X., Zhang D., Guo C., *et al.* AutoCTS: Automated Correlated Time Series Forecasting // Proc. VLDB Endow. 2021. Vol. 15, no. 4. P. 971–983. DOI: 10.14778/3503585.3503604.

148. Xu Y., Zhu S., Yang S., *et al.* Learning with Non-Convex Truncated Losses by SGD // Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019. Vol. 115 / ed. by A. Globerson, R. Silva. AUAI Press, 2019. P. 701–711. (Proceedings of Machine Learning Research).

149. Yang C., Cheung Y., Ding J., Tan K.C. Concept Drift-Tolerant Transfer Learning in Dynamic Environments // IEEE Trans. Neural Networks Learn. Syst. 2022. Vol. 33, no. 8. P. 3857–3871. DOI: 10.1109/TNNLS.2021.3054665.

150. Yi B., Sidiropoulos N.D., Johnson T., *et al.* Online Data Mining for Co-Evolving Time Sequences // Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000 / ed. by D.B. Lomet, G. Weikum. IEEE Computer Society, 2000. P. 13–22. DOI: 10.1109/ICDE.2000.839383.

151. Yi X., Zheng Y., Zhang J., Li T. ST-MVL: Filling Missing Values in Geo-Sensory Time Series Data // Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016 / ed. by S. Kambhampati. IJCAI/AAAI Press, 2016. P. 2704–2710. DOI: 10.5555/3060832.3060999.

152. Ying C., Klein A., Christiansen E., *et al.* NAS-Bench-101: Towards Reproducible Neural Architecture Search // Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Vol. 97 / ed. by K. Chaudhuri, R. Salakhutdinov. PMLR, 2019. P. 7105–7114. (Proceedings of Machine

Learning Research). URL: http://proceedings.mlr.press/v97/ying19a.html.

153. 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.

154. Yu H., Rao N., Dhillon I.S. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction // Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain / ed. by D.D. Lee, M. Sugiyama, U. von Luxburg, *et al.* 2016. P. 847–855. URL: https://proceedings.neurips.cc/paper/2016/hash/85422afb467e9456013a2a51d4dff702-Abstract.html.

155. Yu K., Pang Z., Gidlund M., *et al.* REALFLOW: Reliable Real-Time Flooding-Based Routing Protocol for Industrial Wireless Sensor Networks // Int. J. Distributed Sens. Networks. 2014. Vol. 10. DOI: 10.1155/2014/936379.

156. Yuan H., Xu G., Yao Z., *et al.* Imputation of Missing Data in Time Series for Air Pollutants Using Long Short-Term Memory Recurrent Neural Networks // Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, UbiComp/ISWC 2018 Adjunct, Singapore, October 08-12, 2018. ACM, 2018. P. 1293–1300. DOI: 10.1145/3267305.3274648.

157. Yurtin A.A. Mean Profile Distance Error. (Accessed: 6.6.2025). https://gitverse.ru/yurtinaa/MPDE.

158. Yurtin A.A. Snippet and Artifici-al Neural Network-based Imputation. (Accessed: 6.6.2025). https://gitverse.ru/yurtinaa/SANNI.

159. Yurtin A.A. Snippet-based Autoencoderfor Time-series Imputation. (Accessed: 6.6.2025). https://gitverse.ru/yurtinaa/SAETI.

160. Yurtin A.A. Time-series Graph-Attention Perfomance Predict model. (Accessed: 6.6.2025). https://gitverse.ru/yurtinaa/tsGAP2.

161. Yurtin A.A., Zymbler M.L. SANNI: Online Imputation of Missing Values in Multivariate Time Series Based on Deep Learning and Behavioral Patterns // Lobachevskii Journal of Mathematics. 2024. Vol. 45, no. 11. P. 5948–5966. DOI: 10.1134/S1995080224606854.

162. Zela A., Siems J.N., Zimmer L., *et al.* Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks // The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. 2022. URL: https://openreview.net/forum?id=OnpFa95RVqs.

163. Zhang J., Yin P. Multivariate Time Series Missing Data Imputation Using Recurrent Denoising Autoencoder // 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). 2019. P. 760–764. DOI: 10.1109/BIBM47256.2019.8982996.

164. Zhao B., Lu H., Chen S., *et al.* Convolutional neural networks for time series classification // Journal of Systems Engineering and Electronics. 2017. Vol. 28, no. 1. P. 162–169. DOI: 10.21629/JSEE.2017.01.18.

165. Zhao L., Song Y., Zhang C., *et al.* T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction // IEEE Trans. Intell. Transp. Syst. 2020. Vol. 21, no. 9. P. 3848–3858. DOI: 10.1109/TITS.2019.2935152.

166. 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.

167. Zhu H., Pang Z., Xie B., Bag G. IETF IoT based wireless communication for latency-sensitive use cases in building automation // 25th IEEE International Symposium on Industrial Electronics, ISIE 2016,

Santa Clara, CA, USA, June 8-10, 2016. IEEE, 2016. P. 1168–1173. DOI: 10.1109/ISIE.2016.7745060.

168. Zymbler M., Goglachev A. Fast summarization of long time series with graphics processor // Mathematics. 2022. Vol. 10, no. 10. P. 1781. DOI: 10.3390/math10101781.

Приложение А. Результаты вычислительных экспериментов для метода SANNI

Табл. А.1. Ошибка в сценарии Blackout, RMSE $\times 10^{-3}$

Метод		Набор данных													
		Категория А: Активности						*Категория Б: Сезонности и/или цикличность*					*Категория В: Стохастические*		
Тип	Название	Electricity	Madrid	PAMAP	NREL	WalkRun	*Среднее*	BAFU	MAREL	Meteo	Saaleaue	*Среднее*	BTC	Soccer	*Среднее*
Аналит.	CDRec	253 (6)	213 (6)	118 (6)	194 (5)	91 (5)	174 (5)	78 (3)	194 (8)	130 (3)	**92 (1)**	124 (2)	4 (1-3)	59 (6)	32 (3)
	DynaMMO	349 (9)	286 (9)	252 (10)	378 (9-10)	305 (10)	314 (10)	230 (10)	332 (10)	299 (8)	502 (9)	341 (10)	4 (1-3)	78 (7)	41 (5)
	ROSL	266 (7)	215 (7)	122 (8)	195 (6)	110 (6)	182 (6)	74 (2)	191 (7)	110 (2)	99 (3)	**118 (1)**	4 (1-3)	53 (4)	28 (2)
Нейросетевые	BRITS	305 (8)	232 (8)	120 (7)	253 (8)	178 (8)	218 (7)	172 (8)	239 (9)	380 (9)	263 (8)	264 (8)	17 (6)	130 (8)	74 (8)
	GP-VAE	447 (10)	129 (3)	102 (3)	378 (9-10)	112 (7)	234 (9)	**67 (1)**	174 (3)	444 (10)	578 (10)	316 (9)	9 (4)	240 (9)	124 (9)
	M-RNN	231 (5)	294 (10)	145 (9)	241 (7)	245 (9)	231 (8)	184 (9)	171 (2)	227 (7)	182 (7)	191 (7)	155 (10)	324 (10)	240 (10)
	NAOMI	142 (2)	37 (2)	111 (4-5)	**138 (1)**	**84 (1-4)**	102 (2)	155 (7)	**170 (1)**	**87 (1)**	94 (2)	126 (3)	61 (9)	**27 (1)**	44 (6)
	SAITS	169 (4)	164 (4-5)	**96 (1-2)**	144 (3-4)	**84 (1-4)**	131 (4)	99 (5-6)	189 (5-6)	164 (5-6)	120 (4-5)	143 (5-6)	35 (7-8)	45 (3)	40 (4)
	Transformer	156 (3)	164 (4-5)	**96 (1-2)**	144 (3-4)	**84 (1-4)**	129 (3)	99 (5-6)	189 (5-6)	164 (5-6)	120 (4-5)	143 (5-6)	35 (7-8)	57 (5)	46 (7)
	SANNI	**84 (1)**	**34 (1)**	111 (4-5)	142 (2)	**84 (1-4)**	**91 (1)**	88 (4)	179 (4)	147 (4)	151 (6)	141 (4)	10 (5)	42 (2)	**26 (1)**

Табл. А.2. Ошибка в сценарии MCAR, RMSE $\times 10^{-3}$

Метод		Набор данных													
		Категория А: Активности						*Категория Б: Сезонности и/или цикличность*					*Категория В: Стохастические*		
Тип	Название	Electricity	Madrid	PAMAP	NREL	WalkRun	*Среднее*	BAFU	MAREL	Meteo	Saaleaue	*Среднее*	BTC	Soccer	*Среднее*
Аналитические	CDRec	106 (12)	103 (14)	137 (14)	120 (9)	177 (15)	129 (13)	75 (12)	358 (16)	78 (7-9)	119 (13)	158 (15)	60 (7-8)	125 (12)	92 (12)
	DynaMMO	100 (7)	71 (6)	57 (6)	111 (5)	127 (7)	93 (5)	39 (4)	147 (5)	73 (4)	85 (5)	86 (4)	60 (7-8)	79 (5)	70 (5)
	GROUSE	131 (15)	427 (16)	551 (16)	512 (16)	303 (16)	385 (16)	362 (16)	241 (15)	199 (16)	149 (16)	238 (16)	146 (16)	183 (16)	164 (16)
	ORBITS	105 (10-11)	90 (10)	82 (11)	114 (6-7)	174 (14)	113 (11)	109 (15)	192 (11)	85 (11)	104 (9)	122 (12)	67 (11-13)	113 (10-11)	90 (11)
	ROSL	110 (13)	91 (11)	72 (9-10)	141 (12)	144 (8)	112 (10)	59 (8)	169 (8)	78 (7-9)	88 (7)	98 (6)	76 (14)	96 (6)	86 (10)
	SoftImp.	103 (8)	76 (7)	70 (8)	116 (8)	156 (10)	104 (7)	65 (10)	188 (10)	80 (10)	100 (8)	108 (9)	61 (9)	106 (7)	84 (8)
	SVDImp.	105 (10-11)	80 (8-9)	101 (12)	123 (10)	161 (11)	114 (12)	62 (9)	216 (14)	77 (5-6)	107 (11-12)	116 (11)	58 (6)	110 (9)	84 (9)
	SVT	95 (6)	80 (8-9)	72 (9-10)	87 (4)	154 (9)	98 (6)	74 (11)	180 (9)	78 (7-9)	75 (4)	102 (8)	57 (5)	108 (8)	82 (7)
	TeNMF	104 (9)	93 (12)	386 (15)	131 (11)	166 (13)	176 (15)	52 (7)	214 (13)	77 (5-6)	105 (10)	112 (10)	43 (2)	113 (10-11)	78 (6)
Нейросетевые	BRITS	116 (14)	96 (13)	35 (2)	182 (14)	120 (6)	110 (9)	86 (14)	201 (12)	168 (15)	74 (3)	132 (14)	67 (11-13)	139 (13)	103 (13)
	GP-VAE	91 (5)	68 (5)	61 (7)	225 (15)	78 (4)	105 (8)	41 (5-6)	155 (6-7)	114 (13)	86 (6)	99 (7)	67 (11-13)	149 (14)	108 (14)
	M-RNN	167 (16)	105 (15)	129 (13)	168 (13)	165 (12)	147 (14)	77 (13)	155 (6-7)	132 (14)	134 (15)	124 (13)	137 (15)	164 (15)	150 (15)
	NAOMI	76 (2-3)	49 (4)	49 (5)	114 (6-7)	80 (5)	74 (4)	41 (5-6)	128 (4)	86 (12)	107 (11-12)	90 (5)	65 (10)	**9 (1)**	37 (3)
	SAITS	79 (4)	45 (2-3)	45 (3-4)	**54 (1-2)**	**61 (1-2)**	57 (3)	**24 (1-2)**	101 (1-3)	72 (2-3)	**63 (1-2)**	**65 (1-2)**	53 (3-4)	17 (2)	**35 (1)**
	Transformer	76 (2-3)	45 (2-3)	45 (3-4)	**54 (1-2)**	**61 (1-2)**	56 (2)	**24 (1-2)**	101 (1-3)	72 (2-3)	**63 (1-2)**	**65 (1-2)**	53 (3-4)	18 (3)	36 (2)
	SANNI	**56 (1)**	**29 (1)**	**33 (1)**	71 (3)	76 (3)	**53 (1)**	25 (3)	101 (1-3)	**63 (1)**	125 (14)	78 (3)	**39 (1)**	76 (4)	58 (4)

Табл. А.3. Ошибка в сценарии Overlap, RMSE $\times 10^{-3}$

Метод		Набор данных													
		Категория А: Активности						*Категория Б: Сезонности и/или цикличность*					*Категория В: Стохастические*		
Тип	Название	Electricity	Madrid	PAMAP	NREL	WalkRun	*Среднее*	BAFU	MAREL	Meteo	Saaleaue	*Среднее*	BTC	Soccer	*Среднее*
Аналитические	CDRec	104 (8)	99 (9)	96 (9)	150 (6-7)	163 (12-13)	122 (6)	56 (5-6)	248 (12)	50 (7)	94 (10)	112 (8)	15 (2)	147 (8)	81 (4)
	DynaMMO	117 (11-12)	82 (2)	88 (8)	140 (4)	140 (8)	113 (4)	48 (2-3)	313 (16)	47 (3)	83 (5)	123 (10)	31 (7)	133 (7)	82 (5)
	GROUSE	112 (10)	317 (14)	254 (13)	447 (16)	284 (16)	283 (16)	225 (15)	243 (9)	268 (15)	147 (14)	221 (14)	137 (15)	173 (10)	155 (13)
	ORBITS	96 (5)	89 (4)	401 (14)	162 (9)	168 (14)	183 (13)	56 (5-6)	209 (5)	76 (11)	97 (11)	110 (6)	77 (12)	125 (3)	101 (9)
	ROSL	110 (9)	96 (6-7)	65 (4)	150 (6-7)	102 (5)	105 (2)	58 (7)	220 (6)	72 (10)	67 (4)	104 (5)	39 (9)	129 (5-6)	84 (6)
	SoftImp.	100 (6)	83 (3)	84 (6)	151 (8)	150 (9)	114 (5)	**46 (1)**	222 (7)	55 (8)	93 (9)	104 (4)	28 (6)	**117 (1)**	72 (2)
	SVDImp.	103 (7)	96 (6-7)	170 (12)	149 (5)	157 (10)	135 (10)	48 (2-3)	252 (13)	49 (5-6)	91 (8)	110 (7)	47 (10)	129 (5-6)	88 (7)
	SVT	94 (4)	91 (5)	86 (7)	164 (11)	90 (4)	105 (3)	52 (4)	227 (8)	48 (4)	62 (3)	97 (3)	**11 (1)**	118 (2)	**64 (1)**
	TeNMF	117 (11-12)	174 (12)	591 (15)	166 (12)	158 (11)	241 (14)	658 (16)	245 (11)	49 (5-6)	85 (6-7)	259 (16)	36 (8)	214 (11)	125 (10)
Нейросетевые	BRITS	139 (13)	141 (11)	102 (10)	163 (10)	103 (6)	130 (7)	96 (13)	291 (15)	180 (14)	105 (12)	168 (13)	70 (11)	223 (12)	146 (12)
	GP-VAE	283 (16)	97 (8)	75 (5)	267 (15)	176 (15)	180 (12)	93 (12)	186 (2)	90 (12)	85 (6-7)	114 (9)	90 (13)	249 (14)	170 (14)
	M-RNN	199 (15)	136 (10)	105 (11)	225 (13)	108 (7)	155 (11)	70 (11)	**177 (1)**	156 (13)	203 (15)	152 (12)	131 (14)	259 (15)	195 (15)
	NAOMI	143 (14)	198 (13)	603 (16)	236 (14)	163 (12-13)	269 (15)	139 (14)	244 (10)	311 (16)	204 (16)	224 (15)	180 (16)	676 (16)	428 (16)
	SAITS	86 (3)	365 (15-16)	61 (2-3)	112 (2-3)	**37 (1-2)**	132 (9)	64 (8-9)	200 (3-4)	**43 (1-2)**	**56 (1-2)**	**91 (1-2)**	26 (4-5)	128 (4)	77 (3)
	Transformer	83 (2)	365 (15-16)	61 (2-3)	112 (2-3)	**37 (1-2)**	132 (8)	64 (8-9)	200 (3-4)	**43 (1-2)**	**56 (1-2)**	**91 (1-2)**	26 (4-5)	167 (9)	96 (8)
	SANNI	**81 (1)**	**68 (1)**	**59 (1)**	**67 (1)**	62 (3)	**67 (1)**	65 (10)	258 (14)	62 (9)	122 (13)	127 (11)	23 (3)	244 (13)	134 (11)

Табл. А.4. Ошибка в сценарии Disjoint, RMSE $\times 10^{-3}$

Метод		Набор данных													
		Категория А: Активности						*Категория Б: Сезонности и/или цикличность*					*Категория В: Стохастические*		
Тип	Название	Electricity	Madrid	PAMAP	NREL	WalkRun	*Среднее*	BAFU	MAREL	Meteo	Saaleaue	*Среднее*	BTC	Soccer	*Среднее*
Аналитические	CDRec	93 (4)	89 (9)	159 (12)	146 (9-10)	127 (13)	123 (9)	53 (6)	538 (16)	52 (5-7)	96 (11)	185 (14)	12 (2)	140 (9)	76 (7)
	DynaMMO	116 (12)	84 (5-6)	97 (9)	127 (4)	113 (9)	107 (7)	46 (4)	227 (7)	50 (3-4)	71 (6)	98 (5)	13 (3)	129 (7)	71 (4)
	GROUSE	112 (11)	353 (16)	171 (14)	442 (16)	270 (16)	270 (15)	210 (16)	230 (9)	271 (16)	149 (15)	215 (16)	127 (16)	155 (11)	141 (12-13)
	ORBITS	96 (6)	90 (10)	287 (15)	148 (11)	129 (14)	150 (13)	56 (7-9)	211 (6)	71 (11)	101 (13)	110 (8)	45 (12)	122 (5)	84 (9)
	ROSL	111 (10)	91 (11)	72 (5)	141 (6-7)	85 (5-6)	100 (5)	91 (12)	238 (10)	60 (10)	59 (4)	112 (10)	14 (4)	112 (2)	**63 (1-2)**
	SoftImp.	99 (7)	85 (7)	77 (7)	141 (6-7)	118 (10)	104 (6)	45 (2-3)	229 (8)	57 (9)	92 (10)	106 (6)	16 (5)	117 (3-4)	66 (3)
	SVDImp.	102 (8)	86 (8)	161 (13)	146 (9-10)	122 (11)	123 (10)	47 (5)	255 (11)	52 (5-7)	91 (9)	111 (9)	18 (7)	128 (6)	73 (6)
	SVT	95 (5)	84 (5-6)	74 (6)	145 (8)	47 (3)	89 (4)	**43 (1)**	206 (3)	55 (8)	55 (3)	**90 (1)**	**9 (1)**	117 (3-4)	**63 (1-2)**
	TeNMF	104 (9)	96 (12)	115 (10)	163 (12)	125 (12)	121 (8)	45 (2-3)	259 (12)	52 (5-7)	76 (7)	108 (7)	30 (9)	134 (8)	82 (8)
Нейросетевые	BRITS	121 (13)	125 (13)	144 (11)	136 (5)	106 (8)	126 (11)	96 (13)	361 (15)	170 (14)	81 (8)	177 (13)	27 (8)	364 (16)	196 (16)
	GP-VAE	228 (16)	78 (4)	69 (4)	252 (15)	171 (15)	160 (14)	179 (15)	305 (14)	94 (13)	98 (12)	169 (12)	67 (14)	269 (15)	168 (15)
	M-RNN	127 (14)	144 (14)	92 (8)	197 (13)	85 (5-6)	129 (12)	63 (11)	**172 (1)**	87 (12)	60 (5)	96 (4)	99 (15)	183 (13)	141 (12-13)
	NAOMI	178 (15)	255 (15)	310 (16)	238 (14)	86 (7)	213 (16)	152 (14)	201 (2)	206 (15)	182 (16)	185 (15)	56 (13)	241 (14)	148 (14)
	SAITS	90 (2)	60 (2-3)	55 (2-3)	115 (2-3)	**30 (1-2)**	70 (2-3)	56 (7-9)	210 (4-5)	**47 (1-2)**	**51 (1-2)**	91 (2-3)	35 (10-11)	**109 (1)**	72 (5)
	Transformer	92 (3)	60 (2-3)	55 (2-3)	115 (2-3)	**30 (1-2)**	70 (2-3)	56 (7-9)	210 (4-5)	**47 (1-2)**	**51 (1-2)**	91 (2-3)	35 (10-11)	144 (10)	90 (11)
	SANNI	**68 (1)**	**50 (1)**	**49 (1)**	**65 (1)**	51 (4)	**57 (1)**	58 (10)	281 (13)	50 (3-4)	142 (14)	133 (11)	17 (6)	160 (12)	88 (10)

Приложение В. Результаты вычислительных экспериментов для функции потерь MPDE

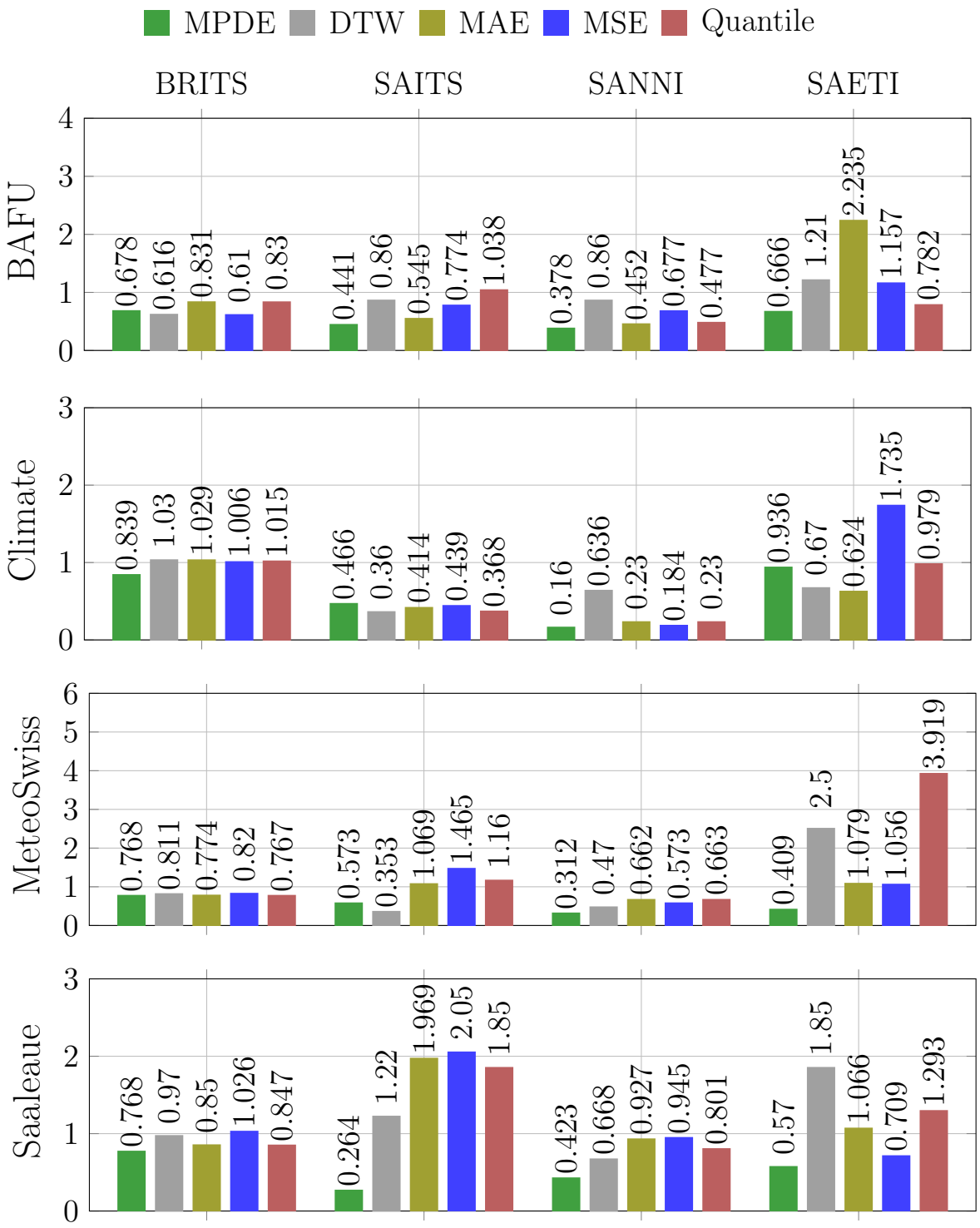


Рис. В.1. Результаты экспериментов для данных категории «Сезонность и/или цикличность», RMSE

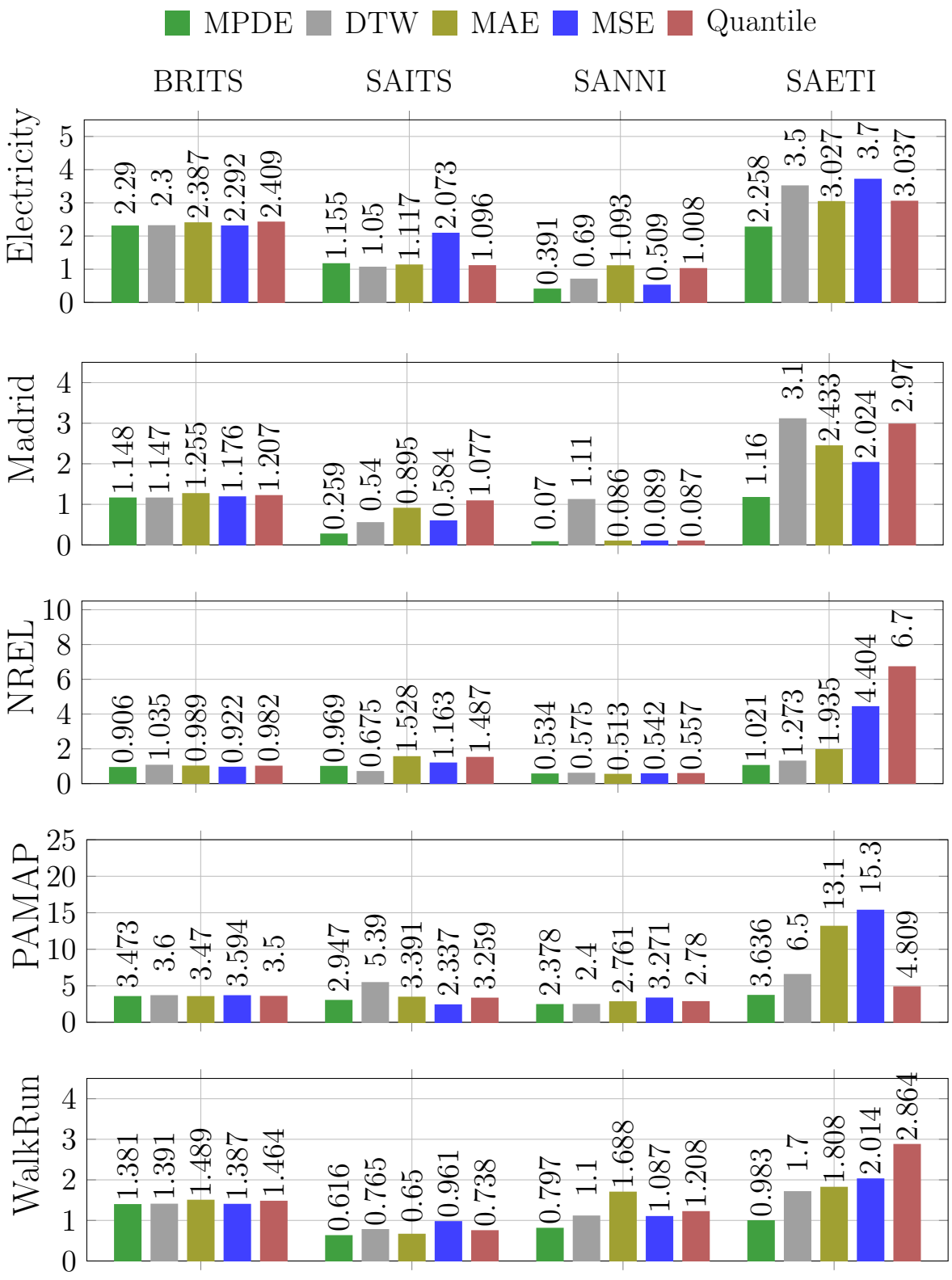


Рис. В.2. Результаты экспериментов для данных категории «Активности», RMSE

Приложение С. Структура входных данных метода tsGAP

В табл. С.1, С.2, С.3 приведено подробное описание входных данных метода tsGAP, которые использовались для проведения вычислительных экспериментов. Таблица С.1 представляет собой перечень функций активации, используемых в анализируемых нейронных моделях метода tsGAP. Каждая строка таблицы соответствует одной функции активации.

Табл. С.1. Типы функций активации анализируемых нейросетевых моделей

№ п/п	Название	Описание
1.	NONE	Без активации
2.	ReLU	Обнуление отрицательных значений
3.	Sigmoid	Сигмоида
4.	Tanh	Гиперболический тангенс
5.	Leaky ReLU	Модифицированная ReLU с малым коэффициентом для отрицательных значений

Таблица С.2 представляет собой описание элементов входного вектора параметров обучения, поступающего на вход метода tsGAP. Каждая строка таблицы соответствует одному параметру, включенному в вектор.

Табл. С.2. Состав входного вектора параметров обучения

№ п/п	Параметр
1.	Длина подпоследовательности
2.	Количество координат
3.	Скорость обучения
4.	Объем доступной видеопамяти (GB)
5.	Пропускная способность памяти (GB/s)
6.	Тензорные ядра
7.	Производительность (TFLOPS)
8.	CUDA-ядра
9.	Тактовая частота GPU (GHz)
10.	Версия архитектуры (CUDA Capability)

В табл. С.3 представлены слои, используемые в целевых моделях метода tsGAP, с указанием их функционального назначения и основных параметров. Каждая строка таблицы соответствует конкретному слою и содержит его порядковый номер, название, описание операции, которую слой реализует и параметры, определяющие конфигурацию слоя.

Табл. С.3. Типы слоев целевой моделей

№ п/п	Название	Описание	Параметры слоя	
			Первый	Второй
1.	NONE	Отсутствие слоя	отсутствует	
2.	DENSE	Полносвязный слой	количество нейронов	отсутствует
3.	RNNRESULT	Обработка результата рекуррентного слоя	отсутствует	
4.	RNN	Рекуррентный слой	размер скрытого состояния	отсутствует
5.	LSTM	Рекуррентный слой LSTM	размер скрытого состояния	отсутствует
6.	GRU	Рекуррентный слой GRU	размер скрытого состояния	отсутствует
7.	CONV1D	Одномерный сверточный слой	число фильтров	размер ядра
8.	MAXPOOLING1D	Одномерный MaxPooling	размер ядра	шаг
9.	VECTTOMATRIX	Преобразование вектора в матрицу	первое измерение	второе измерение
10.	TRANSPOSE	Перестановка размерностей входного тензора	количество строк	количество столцов
11.	FLATTEN	Преобразование многомерной матрицы в вектор	отсутствует	
12.	TIMESERIESIMPUTE	Входной слой, принимающий подпоследовательности	длина	количество измерений
13.	NANINIT	Инициализация пропусков	значение	отсутствует