A METHOD FOR FORECASTING THE ERROR AND TRAINING TIME OF NEURAL NETWORKS FOR MULTIVARIATE TIME SERIES IMPUTATION

A. A. Yurtin

South Ural State University (National Research University),
454080, Chelyabinsk, Russia

DOI: 10.24412/2073-0667-2025-3-72-95
EDN: XLSZLH

The article presents a neural network-based method called tsGAP2, designed for predicting the error and training time of neural network models used for imputing missing values in multivariate time series. The input data for the method are neural network represented as a directed acyclic graphs, where nodes correspond to layers and edges represent connections between them. The method involves three components: an Autoencoder, which transforms the graph-based representation of the model into a compact vector form; an Encoder, which encodes the hyperparameters and characteristics of the computational device; and an Aggregator, which combines the vector representations to generate the prediction. Training of the tsGAP2 neural network model is carried out using a composite loss function, defined as a weighted sum of multiple components. Each component evaluates different aspects of the tsGAP2 model's output, including the correctness of the decoded neural network model from the vector representation, the prediction of the model's error, and its training time. For the study, a search space comprising 200 different architectures was constructed. During the experiments, 12,000 training runs were conducted on time series from various application domains. The experimental results demonstrate that the proposed method achieves high accuracy in predicting the target model's error: the average error, measured using SMAPE, is 4.4 %, which significantly outperforms existing alternative approaches, which show an average error of 27.6 %. The average prediction error for training time was 8.8 %, also significantly better than existing methods, which show an error of 61.6 %.

Key words: time series, missing value imputation, neural network models, autoencoder, graph neural networks, attention mechanism, performance prediction, neural architecture search.

References

1. Aydin S. Time series analysis and some applications in medical research // Journal of Mathematics and Statistics Studies. 2022. V. 3. N 2. P. 31–36. DOI: 10.32996/JMSS.

2. Voevodin V. V., Stefanov K. S. Development of a portable software solution for monitoring and analyzing the performance of supercomputer applications // Numerical Methods and Programming. 2023. V. 24. P. 24–36. DOI: 10.26089/NumMet.v24r103.

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

The work was carried out with financial support from the Russian Science Foundation (grant N 23-21-00465).

4. Gromov V. A., Lukyanchenko P. P., Beschastnov Yu. N., Tomashchuk K. K. Time Series Structure Analysis of the Number of Law Cases // Proceedings in Cybernetics. 2022. N 4 (48). P. 37–48.

5. Kazijevs M., Samad M. D. Deep imputation of missing values in time series health data: A review with benchmarking // J. Biomed. Informatics. 2023. V. 144. P. 104440. DOI: 10.1016/J.JBI.2023.104440.

6. Elsken T., Metzen J. H., Hutter F. Neural Architecture Search: A Survey // J. Mach. Learn. Res. 2019. V. 20. N 55. P. 1–21. [Electron. res.]: https://jmlr.org/papers/v20/18-598.html.

7. Wozniak A. P., Milczarek M., Wozniak J. MLOps Components, Tools, Process, and Metrics: A Systematic Literature Review // IEEE Access. 2025. V. 13. P. 22166–22175. DOI: 10.1109/ACCESS.2025.3534990.

8. Weights & Biases: Machine learning experiment tracking, dataset versioning, and model management. [El. Res.]: https://wandb.ai/. Access date: 2025-06-11.

9. Bergstra J., Bengio Y. Random search for hyper-parameter optimization // J. Mach. Learn. Res. 2012. V. 13. P. 281–305.

10. Dong X., Yang Y. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search // 8th Int. Conf. on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020. [Electron. res.]: https://openreview.net/forum?id=HJxyZkBKDr.

11. Ding Y., Huang Z., Shou X., Guo Y., Sun Y., Gao J. Architecture-Aware Learning Curve Extrapolation via Graph Ordinary Differential Equation // AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, Feb. 25 — Mar. 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.

12. timeseries Graph Attention Performance Predict. [El. Res.]: https://gitverse.ru/yurtinaa/tsGAP2. Access date: 2025-05-03.

13. Gawlikowski J., Tassi C. R. N., Ali M., Lee J., Humt M., Feng J., Kruspe A., Triebel R., Jung P., Roscher R., Shahzad M., Yang W., Bamler R., Zhu X. X. A survey of uncertainty in deep neural networks // Artif. Intell. Rev. 2023. V. 56. N 1. P. 1513–1589. ISSN: 1573–7462. DOI: 10.1007/s10462-023-10562-9.

14. Zela A., Siems J. N., Zimmer L., Lukasik J., Keuper M., Hutter F. Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks // The Tenth Int. Conf. on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022. [Electron. res.]: https://openreview.net/forum?id=0npFa95RVqs.

15. Titsias M. Variational Learning of Inducing Variables in Sparse Gaussian Processes // Proc. of the Twelfth Int. Conf. on Artificial Intelligence and Statistics. / ed. by D. van Dyk, M. Welling. Hilton Clearwater Beach Resort, Clearwater Beach, Florida, USA: PMLR, 16–18 Apr. 2009. V. 5. P. 567–574. [Electron. res.]: https://proceedings.mlr.press/v5/titsias09a.html.

16. Ying C., Klein A., Christiansen E., Real E., Murphy K., Hutter F. NAS-Bench-101: Towards Reproducible Neural Architecture Search // Proc. of the 36th Int. Conf. on Machine Learning, ICML 2019, June 9–15, Long Beach, California, USA / ed. by K. Chaudhuri, R. Salakhutdinov. PMLR, 2019. V. 97. P. 7105–7114. [Electron. res.]: http://proceedings.mlr.press/v97/ying19a.html.

17. White C., Neiswanger W., Savani Y. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search // Thirty-Fifth AAAI Conf. on Artificial Intelligence, AAAI 2021, IAAI 2021, EAAI 2021, Virtual Event, Feb. 2–9, 2021. AAAI Press, 2021. P. 10293–10301. DOI: 10.1609/AAAI.V35I12.17233.

18. White C., Zela A., Ru R., Liu Y., Hutter F. How powerful are performance predictors in neural architecture search? // Adv. Neural Inf. Process. Syst. 2021. V. 34. P. 28454–28469.

19. Snoek J., Rippel O., Swersky K., Kiros R., Satish N., Sundaram N., Patwary M., Prabhat, Adams R. P. Scalable Bayesian Optimization Using Deep Neural Networks // Proc. of the 32nd Int. Conf. on Machine Learning (ICML). Lille, France: PMLR, 2015. V. 37. P. 2171–2180.

20. Springenberg J. T., Klein A., Falkner S., Hutter F. Bayesian Optimization with Robust Bayesian Neural Networks // Adv. Neural Inf. Process. Syst / ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett. V. 29.

21. Wu X., Zhang D., Guo C., He C., Yang B., Jensen C. S. AutoCTS: Automated Correlated Time Series Forecasting // Proc. VLDB Endow. 2021. V. 15. N 4. P. 971–983. DOI: 10.14778/3503585.3503604.

22. Wang C., Chen X., Wu C., Wang H. AutoTS: Automatic Time Series Forecasting Model Design Based on Two-Stage Pruning // arXiv preprint: abs/2203.14169. DOI: 10.48550/arXiv.2203.14169.

23. Velickovic P., Cucurull G., Casanova A., Romero A., Li'o P., Bengio Y. Graph Attention Networks // 6th Int. Conf. on Learning Representations, ICLR 2018, Vancouver, Canada, April 30 — May 3, 2018. 2018. [Electron. res.]: https://openreview.net/forum?id=rJXMpikCZ.

24. Clevert D., Unterthiner T., Hochreiter S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) // 4th Int. Conf. on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016 / ed. by Y. Bengio, Y. LeCun. 2016. [Electron. res.]: http://arxiv.org/abs/1511.07289.

25. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. V. 6. N 2. P. 107–116. DOI: 10.1142/S0218488598000094.

26. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // 2016 IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, USA. IEEE Computer Society. 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.

27. Srivastava N., Hinton G. E., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting // J. Mach. Learn. Res. 2014. V. 15. N 1. P. 1929–1958. DOI: 10.5555/2627435.2670313.

28. Mao A., Mohri M., Zhong Y. Cross-Entropy Loss Functions: Theoretical Analysis and Applications // Proc. of the 40th Int. Conf. on Machine Learning / ed. by A. Krause. 2023. V. 202. P. 23803–23828.

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

30. Bilenko R. V., Dolganina N. Yu., Ivanova E. V., Rekachinsky A. I. High-performance Computing Resources of South Ural State University // Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. V. 11. N 1. P. 15–30. DOI: 10.14529/cmse220102.

31. BundesAmt Für Umwelt — Swiss Federal Office for the Environment. [El. Res.]: https://www.hydrodaten.admin.ch/. Access date: 2025-05-03.

32. Trindade A., "Electricity Load Diagrams 2011–2014," UCI Machine Learning Repository (2015) [El. Res.]: https://doi.org/10.24432/C58C86. Access date: 2023-05-03.

33. Lozano A. C., Li H., Niculescu-Mizil A., Liu Y., Perlich C., Hosking J. R. M., Abe N. Spatial-temporal causal modeling for climate change attribution // Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Paris, France, June 28 — July 1, 2009 / ed. by J. F. Elder IV, F. Fogelman-Soulié, P. A. Flach, M. J. Zaki. — ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

34. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques // Transp. Res. Part C: Emerg. Technol. 2018. V. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

35. Sheppy M., Beach A., Pless S. NREL RSF Measured Data 2011. [El. Res.]: https://data.openei.org/submissions/358. Access date: 2023-09-03.

36. Snytnikov A. V., Ezrokh Yu. S. Solving Vlasov Equation with Neural Networks // Lobachevskii Journal of Mathematics. 2024. V. 45. P. 3416–3423.

МЕТОД ПРОГНОЗИРОВАНИЯ ОШИБКИ ВРЕМЕНИ ОБУЧЕНИЯ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ ВОССТАНОВЛЕНИЯ МНОГОМЕРНЫХ ВРЕМЕННЫХ РЯДОВ

А. А. Юртин

Южно-Уральский государственный университет
(Национальный исследовательский университет)
454080, Челябинск, Россия

УДК 04.032.26, 004.048
DOI: 10.24412/2073-0667-2025-3-72-95
EDN: XLSZLH

В статье представлен нейросетевой метод tsGAP2, предназначенный для прогнозирования ошибки и времени обучения нейросетевых моделей восстановления пропущенных значений в многомерных временных рядах. Входными данными метода является нейросетевая модель, представленная в виде ориентированного ациклического графа, в которой узлы соответствуют слоям, а дуги — связи между ними. Метод предполагает использование трех компонентов: Автоэнкодера, который преобразует графовое представление модели в компактное векторное, Энкодера, кодирующего гиперпараметры и характеристики вычислительного устройства, и Агрегатора, объединяющего векторные представления и формирующего прогноз. Обучение нейросетевой модели tsGAP2 осуществляется с использованием составной ошибки, представляющей собой взвешенную сумму нескольких компонент. Каждая компонента оценивает различные аспекты выхода модели tsGAP2, включая корректность декодированной из векторного представления нейросетевой модели, прогноз ошибки и времени ее обучения. Для исследования было сформировано пространство поиска, включающее 200 различных архитектур. Во время экспериментов было выполнено 12 000 запусков обучения на временных рядах из различных предметных областей. Результаты экспериментов показывают, что предложенный метод обеспечивает высокую точность прогнозирования ошибки целевой модели: средняя ошибка по мере SMAPE составляет 4.4 %, что значительно превосходит существующие альтернативные подходы, демонстрирующие ошибку в среднем на уровне 27.6 %. Средняя ошибка прогноза времени составила 8.8 %, что значительно превосходит существующие альтернативные подходы, демонстрирующие ошибку, равную 61.6 %.

Ключевые слова: временные ряды, восстановление пропущенных значений, нейросетевые модели, автоэнкодер, графовые нейронные сети, механизм внимания, время обучения, ошибка, поиск архитектуры нейросетей.

Введение. В настоящее время в широком спектре предметных областей востребована интеллектуальная обработка многомерных временных рядов: здравоохранение [1], суперкомпьютерные системы [2], интернет вещей [3], юриспруденция [4] и др. Однако реальные данные зачастую содержат пропуски, возникающие по различным причинам. Наличие

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

пропусков существенно осложняет последующую обработку временных рядов. Пропущенные значения могут интерпретироваться как шум или аномалии, что приводит к снижению точности статистических методов и моделей машинного обучения. Кроме того, многие существующие методы требуют полноты данных на входе и не предназначены для работы с пропусками, что делает предварительное восстановление данных важным этапом предварительной обработки.

Среди современных методов восстановления пропусков во временных рядах одним из актуальных направлений считается использование нейросетевых моделей [5]. Благодаря способности выявлять закономерности как в последовательности данных, так и между измерениями одного временного ряда, нейросетевые модели демонстрируют высокую эффективность в задачах анализа и восстановления многомерных временных рядов. Архитектурные решения, применяемые при моделировании временных рядов, охватывают широкий спектр нейросетевых подходов: рекуррентные нейронные сети (например, LSTM и GRU), нейронные сети, включающие механизм внимания (трансформеры) и др. Учет характеристик временных рядов и предметной области при выборе архитектуры позволяет адаптировать модель к типу данных, структуре пропусков и особенностям конкретной задачи, тем самым повышая точность восстановления. Однако задача выбора и поиска подходящей архитектуры нейронной сети остается нетривиальной, поскольку связана с накладными расходами по проведению большого количества вычислительных экспериментов.

Одним из актуальных направлений решения вышеописанной проблемы является поиск архитектуры нейронной сети (Neural Architecture Search, NAS) [6]. Поиск архитектуры осуществляется в рамках заданного пространства поиска (Search Space), которое включает в себя нейросетевые модели, определяемые большим количеством параметров: типами, количеством, порядком соединений, функциями активации слоев и др. Методы NAS позволяют автоматизировать выбор нейронной сети путем прогнозирования ее качества или сужения пространства поиска. Данные методы применяются в современных системах управления жизненным циклом моделей машинного обучения (MLOps) [7], технологии AutoML [7], и в платформах для проведения экспериментов и мониторинга моделей, включая Weights & Biases (wandb) [8]. Однако большинство из них используют традиционные методы перебора, например случайный поиск (Random Search) [9] и байесовская оптимизация (Bayesian Optimization), которые требуют проведения множества дополнительных экспериментов для настройки стратегии поиска. В связи с вышеизложенным, актуальной является задача разработки методов прогнозирования ошибки и времени обучения нейросетевых моделей для восстановления временных рядов, позволяющих оценивать качество нейросетевой модели без необходимости полного обучения каждой из них.

Вклад данной статьи можно сформулировать следующим образом:

1) Предложен метод tsGAP2 (timeseries Graph Attention Performance Predict), решающий задачу прогнозирования ошибки и времени обучения нейросетевых моделей восстановления временных рядов. Для реализации метода используются графовые нейронные сети с механизмом внимания, позволяющие анализировать входную нейросетевую модель, представленную в виде ориентированного ациклического графа. Во время формирования прогноза входные данные обрабатываются следующими компонентами: Автоэнкодером, формирующим векторное представление нейросетевой модели, Энкодером, формирующим векторное представление параметров обучения, и Агрегатором, продуцирующим на основе векторных представлений прогноз модели.

2) Проведена серия экспериментов с пространством поиска, включающим 200 уникальных нейросетевых моделей. Во время экспериментов было произведено 12 000 запусков обучения моделей, решающих задачу восстановления временных рядов из различных предметных областей. Объем проведенных экспериментов сопоставим с аналогичными исследованиями из смежных областей NAS [10–11]. В целях обеспечения воспроизводимости вычислительных экспериментов все исходные коды и наборы данных, используемые в данном исследовании, размещены в открытом репозитории [12]. Результаты вычислительных экспериментов демонстрируют высокую точность прогноза ошибки целевой модели: средняя ошибка менее 4.4 %, что существенно превосходит передовые аналоги, для которых средняя ошибка составляет 27.6 %. Средняя ошибка прогноза времени обучения модели составила 8.8 %, тогда как конкуренты демонстрируют в среднем ошибку на уровне 61.1 %.

Статья организована следующим образом. Раздел 1 содержит краткий обзор близких по тематике работ. В разделе 2 приводятся используемые далее формальные определения и нотация. В разделе 3 представлен новый метод прогнозирования ошибки и времени обучения нейросетевых моделей восстановления временного ряда. Раздел 4 содержит результаты вычислительных экспериментов по исследованию эффективности разработанного метода. В разделе 5 обсуждаются ограничения и практическая применимость предложенного метода в задачах NAS. Заключение содержит сводку полученных результатов и направления будущих исследований.

1. Обзор связанных работ. Для краткости изложения в дальнейшем под *качеством нейросетевой модели* в контексте решаемой задачи будем понимать совокупность двух показателей: ошибка модели и время ее обучения на одной эпохе. Более высокое качество соответствует меньшей ошибке и меньшему времени обучения. В задачах NAS для предсказания качества нейросетевой модели применяются различные виды методов, которые условно можно разделить на три группы: градиентные методы обучения ансамблей, вероятностные подходы и нейросетевые модели, основанные на многослойных персептронах или байесовских нейронных сетях (Bayesian neural networks, BNN) [13]. Рассмотрим каждую группу более подробно.

Для решения задач NAS применяются градиентные методы, например XGBoost, NGBoost, LightGBM и Random Forest [14]. Однако их эффективность может быть ограничена необходимостью настройки гиперпараметров и недостаточной способностью моделировать сложные структурные зависимости, характерные для нейронных моделей.

Гауссовские процессы (Gaussian Processes, GP) применяются в задачах NAS благодаря способности моделировать сложные нелинейные зависимости. В работе [15] для моделирования качества модели применяется вариационный разреженный гауссовский процесс (Variational Sparse Gaussian Process, VSGP), обеспечивающий возможность применения GP-моделей в высокоразмерных пространствах признаков, характерных для описания нейронных моделей. Несмотря на использование вариационных приближений, методы на основе гауссовских процессов остаются вычислительно затратными при применении к большим пространствам поиска, содержащим десятки тысяч возможных нейросетевый моделей и характеризующимся высокой размерностью признакового описания (сотни признаков) [16].

В задачах предсказания качества нейросетевых моделей в рамках NAS в качестве модели прогнозирования используется многослойный перцептрон (Multilayer Perceptron,

MLP) [17]. При наличии информативного признакового описания моделей MLP демонстрирует высокую обобщающую способность и устойчивость к переобучению.

В работе [18] рассматривается применение байесовской линейной регрессии для прогнозирования качества нейросетевых моделей. В качестве входных данных используются векторные представления нейросетевых моделей, включающие их структурные характеристики и соответствующие гиперпараметры. В качестве расширения данного подхода предложен метод DNGO (Deep Networks for Global Optimization) [19], использующий нейронную сеть в качестве преобразователя признаков. Нейросетевые модели и гиперпараметры кодируются с помощью нейронной сети в векторное представление. Полученное представление подается на вход байесовской линейной регрессии, которая предсказывает значения целевой функции.

Метод BOHAMIANN (Bayesian Optimization with Hamiltonian Monte-Carlo Artificial Neural Networks) [20], решает задачу прогноза качества нейросетевой модели с помощью байесовской нейронной сети, обученной с использованием стохастического градиентного гамильтониана Монте-Карло (Stochastic Gradient Hamiltonian Monte-Carlo, SGHMC). Метод BOHAMIANN принимает на вход векторные представления нейросетевой модели и предсказывает распределение значений целевой функции, учитывая как ожидаемое значение, так и степень неопределенности прогноза.

Общим недостатком описанных подходов является отсутствие явного учета связей между слоями нейросетевой модели, поскольку они опираются на векторные представления. Для формирования таких представлений часто применяется one-hot кодирование, при котором каждая операция (свертка, пулинг и др.) и ее позиция в нейросетевой модели кодируются бинарным вектором фиксированной длины. Подобные представления не отражают зависимости между слоями модели, такие как порядок слоев, пропуски (skip connections) и другие сложные связи.

В ряде исследований рассматривается применение методов NAS для задач анализа временных рядов. В частности, AutoCTS (Automated Correlated Time Series) [21] предназначен для автоматизированного построения нейросетевых моделей, способных учитывать пространственно-временные зависимости в данных. Процесс формирования модели в данном методе состоит из двух ключевых этапов. На первом этапе осуществляется поиск оптимальных модулей, которые представляют собой специфические комбинации слоев, учитывающих различные характеристики временных рядов. На втором этапе выполняется оптимизация композиции выбранных модулей для определения наиболее эффективной модели.

Другим примером является метод AutoTS (Automatic Time Series Forecasting) [22], который решает задачу автоматического проектирования нейросетевой модели прогнозирования временных рядов. В рассматриваемом методе пространство поиска модели насчитывает порядка 1.8×10^{21} возможных вариантов. Для повышения вычислительной эффективности поиска AutoTS реализует двухэтапную стратегию сужения пространства. На первом этапе проводится поэтапная оптимизация каждого отдельного модуля. На втором этапе осуществляется сужение множества вариантов внутри модели.

Можно заключить, что существующие методы NAS обладают ограниченной способностью к анализу архитектурных особенностей. В задачах обработки временных рядов такие методы, как правило, ограничиваются специализированными типами слоев и обладают высокоразмерными пространствами поиска.

2. Основные определения и нотация. *Временной ряд (time series)* T длины n (обозначаемой как $|T|$) представляет собой последовательность из n хронологически упорядоченных вещественных значений:

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

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

$$T_{i,m} = \{t_q\}_{q=i}^{i+m-1}, \quad 3 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1.$$

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:

$$\boldsymbol{T} = [\{T^{(k)}\}_{k=1}^{d}]^\intercal, \quad \boldsymbol{T}_{i,m} = [\{T_{i,m}^{(k)}\}_{k=1}^{d}]^\intercal, \quad \boldsymbol{t}_i = [\{t_i^{(k)}\}_{k=1}^{d}]^\intercal.$$

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:

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

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 $\overset{\bullet}{\boldsymbol{S}}_{\boldsymbol{T}}^{m}$:

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

В дальнейшем анализируемую нейросетевую модель мы будем называть целевой. *Целевая нейросетевая модель* может быть формально представлена как ориентированный ациклический граф, в котором вершины соответствуют слоям модели, а дуги представляют собой направленные связи между слоями. Введем набор понятий для формального представления нейросетевой модели.

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

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

Для каждого допустимого элемента $\mathcal{C}$ может быть задано до двух числовых параметров, определяющих его конфигурацию и функциональные характеристики.

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

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

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:

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

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

Связи между слоями нейронной модели задаются матрицей R, каждая строка которой соответствует одной дуге ориентированного графа. 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:

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

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

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

$$\text{slice}_{i:j,k:v} : \mathbb{R}^{b \times q} \to \mathbb{R}^{(j-i+1)\times(v-k+1)},$$

$$\text{slice}_{i:j,k:v}(A) = \begin{bmatrix} A_{i,k} & A_{i,k+1} & \cdots & A_{i,v} \\ A_{i+1,k} & A_{i+1,k+1} & \cdots & A_{i+1,v} \\ \vdots & \vdots & \ddots & \vdots \\ A_{j,k} & A_{j,k+1} & \cdots & A_{j,v} \end{bmatrix}, 1 \leqslant i \leqslant j \leqslant b, \quad 1 \leqslant k \leqslant v \leqslant q,$$

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 $\text{slice}_{:,k:v}$. Результатом такой операции является подматрица, содержащая те же строки, что и исходная матрица, и только те столбцы, номера которых принадлежат 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.

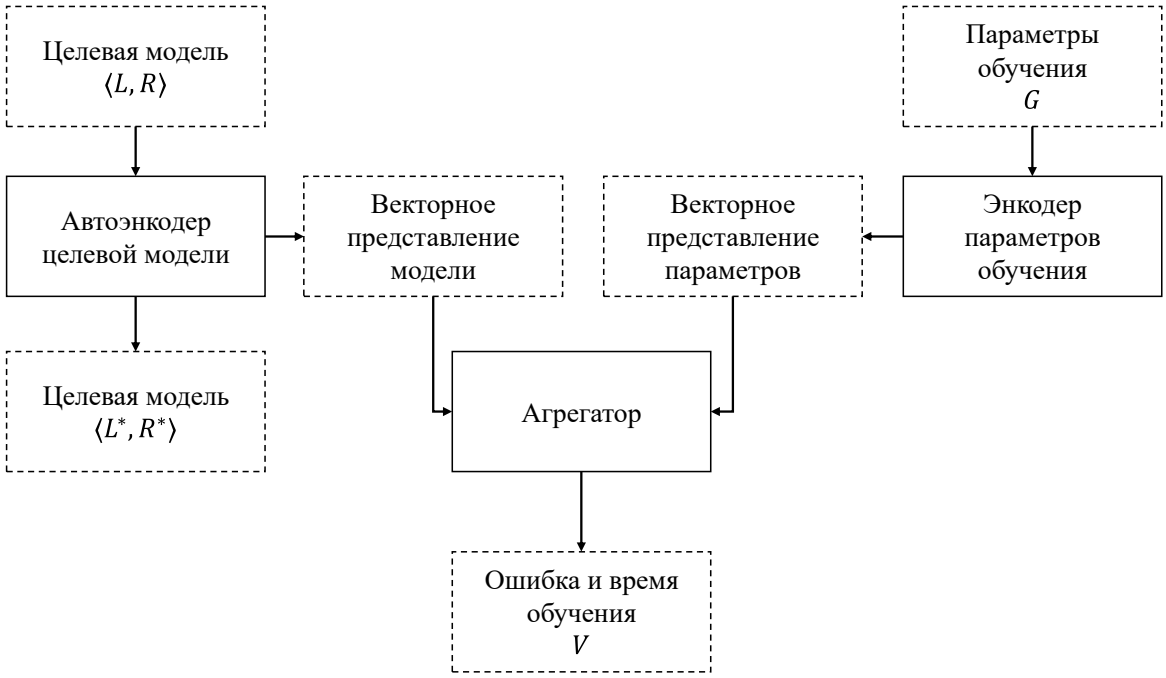

Рис. 1. Метод прогнозирования ошибки и времени обучения нейросетевой модели восстановления многомерного временного ряда

3. Метод прогнозирования ошибки и времени обучения нейросетевой модели. Архитектура предлагаемого метода представлена на рис. 1 и включает следующие компоненты, последовательно обрабатывающие входные данные: Автоэнкодер графового представления, Энкодер параметров обучения и Агрегатор признаков. Автоэнкодер принимает на вход графовое представления нейросетевой модели, представленной двумя элементами: набором слоев L и матрицей связей R. В процессе обработки входных данных Автоэнкодер формирует векторное представление нейросетевой модели, обозначаемое как $Z \in \mathbb{R}^z$. Энкодер получает на вход вектор параметров обучения и преобразует его в векторное представление параметров. Полученные векторные представления целевой модели и параметров обучения передаются на вход Агрегатору, который на выходе продуцирует прогноз в виде вектора из двух значений: ошибки и времени выполнения одной эпохи обучения.

3.1. *Кодирование целевой модели.* В данном разделе рассматривается процесс приведения графового представления целевой модели к векторному, содержащему основную информацию об ее особенностях. Описываются этапы предварительной обработки слоев и связей, включая нормализацию параметров, one-hot кодирование категориальных признаков и формирование входа модели. Рассмотрена структура нейронной сети, которая реализует кодирование целевой модели в векторное представление.

3.1.1. *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:

$$\widehat{p} = \frac{p - p_{\min}}{p_{\max} - p_{min}}, \tag{1}$$

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:

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

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 формально представляться следующим образом:

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

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:

$$\widehat{R}(k,\cdot) = \text{onehot}_\ell(R(k,i)) \cdot \text{onehot}_\ell(R(k,j)), \quad 1 \leqslant k \leqslant \lambda, \quad 1 \leqslant i,j \leqslant \ell. \tag{2}$$

3.1.2. *Формирование векторного представления целевой модели.* Для формирования векторного представления целевой модели используется нейросетевая модель на базе автоэнкодеров, представленная на рис. 2. Нормализованная матрица слоев $\widehat{L}$, исходная матрица связей R и ее нормализованная версия $\widehat{R}$ подаются на вход Автоэнкодера, который представлен на рис. 2, а. Автоэнкодер состоит из четырех подсетей. Первые две подсети представляют собой Энкодер слоев и Энкодер связей, который отвечает за построение векторного представления целевой модели $Z \in \mathbb{R}^z$. Оставшиеся две подсети представляют собой Декодер слоев и Декодер связей, которые восстанавливают графовое представления модели из векторного. В результате работы декодеров формируются декодированные версии матрицы слоев $\widehat{L}^*$, матрицы связей R^* и ее нормализованной версии $\widehat{R}^*$.

Рассмотрим каждую из подсетей более подробно. 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 $2 \cdot \lambda \cdot \ell$ нейронов. Последний слой, состоящий из z нейронов, формирует итоговое векторное представление связей.

Энкодер слоев принимает на вход нормализованную матрицу слоев и матрицу связей. На выходе данной подсети формируется векторное представление слоев. Поскольку целевая нейросетевая модель представляется в виде ориентированного ациклического графа,

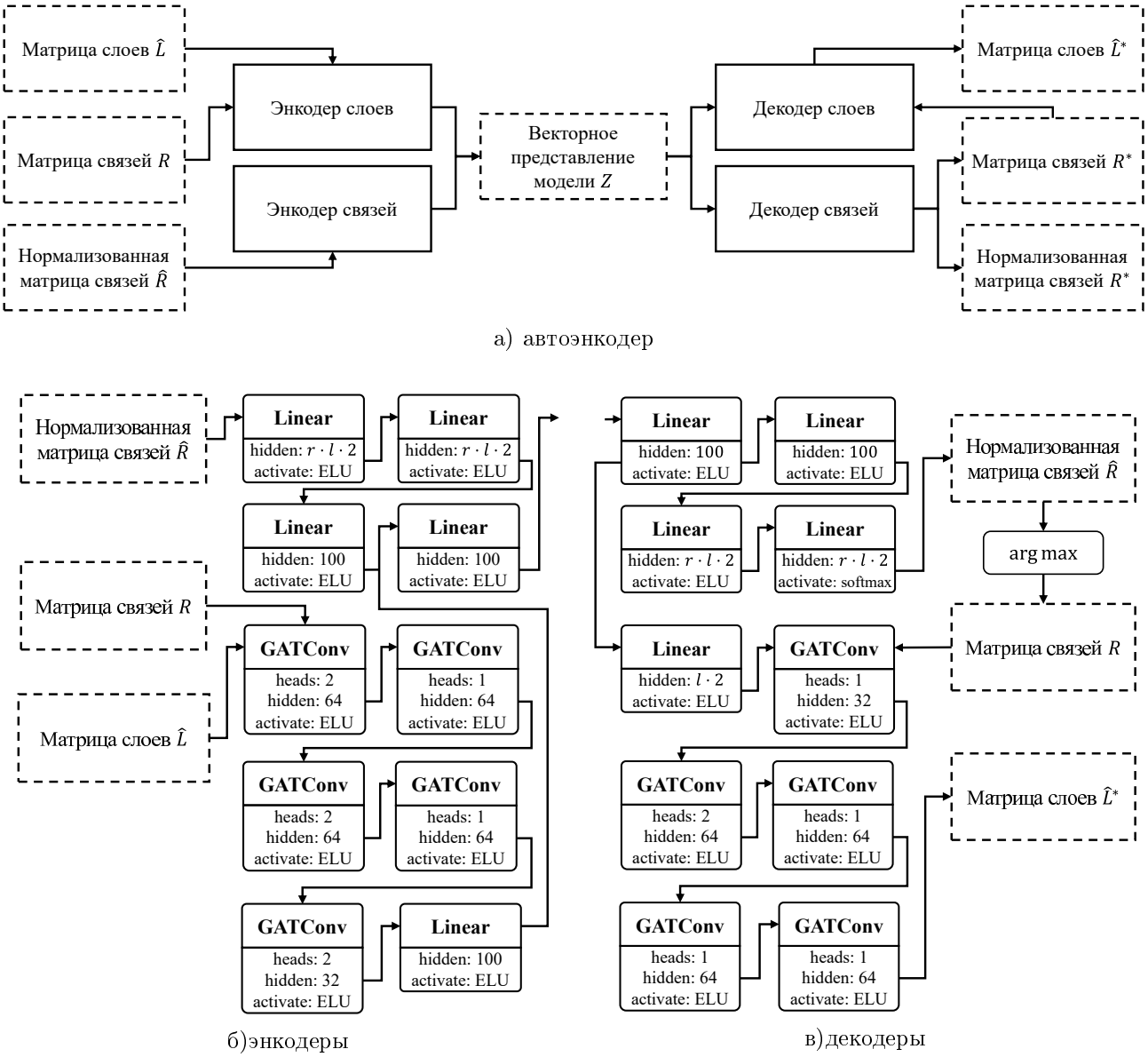

Рис. 2. Модель кодирования целевой модели

для ее обработки используются графовые нейронные сети. В предлагаемом методе применяется модификация графовой сверточной сети с механизмом внимания (Graph Attention Convolution, GATConv) [23]. 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. В первом слое используется 2 головы (head), в последующих по одной голове. Полносвязный слой, состоящий из z нейронов, принимает выход последнего GATConv слоя и формирует векторное представление слоев.

Векторные представления слоев и связей целевой модели объединяются с помощью операции конкатенации. Полученный вектор подается на вход полносвязного слоя с выходной размерностью z, который формирует итоговое векторное представление модели,

обозначаемое как Z. Во всех упомянутых слоях в качестве функции активации используется Exponential Linear Unit (ELU) [24].

Для декодирования целевой 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 $2 \cdot \lambda \cdot \ell$ нейрона. В качестве функции активации для первых двух слоев используется 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 в связи, 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 $32 \cdot \ell$ 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 слоев и имеет размер скрытого представления, равный $c + a + 2$. Во втором GATConv используется 2 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 имеет вид матрицы $O \in \mathbb{R}^{\ell \times (c+a+2)}$, где каждая строка соответствует одному слою 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 вероятность наличия у данного слоя функции активации из набора $\mathcal{A}$. Формально строка декодированной матрицы слоев $\widehat{L}^*$ может быть представлена следующим образом:

$$\begin{gathered}\widehat{L}^*(i,\cdot) = \operatorname{softmax}(\operatorname{slice}_{i,1:c}(O)) \cdot \operatorname{slice}_{i,c:w}(O) \cdot \operatorname{softmax}(\operatorname{slice}_{i,w+1:s}(O)), \\ 1 \leqslant i \leqslant \ell, \quad w = c + 2, \quad s = c + a + 2,\end{gathered} \tag{3}$$

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

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

3.2. *Формирование векторного представления параметров обучения.* На рис. 3 представлена архитектура Энкодера параметров обучения. На вход Энкодеру поступает вектор

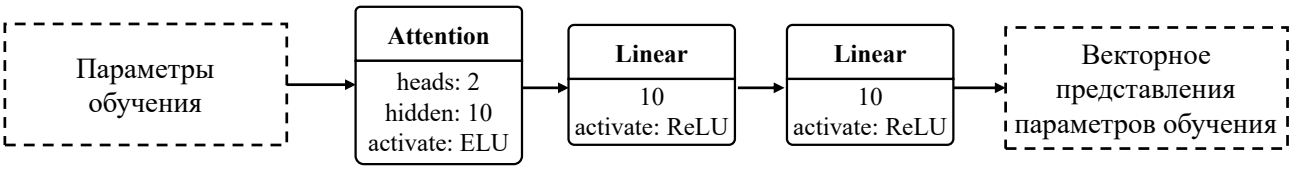

Рис. 3. Модель кодирования параметров обучения

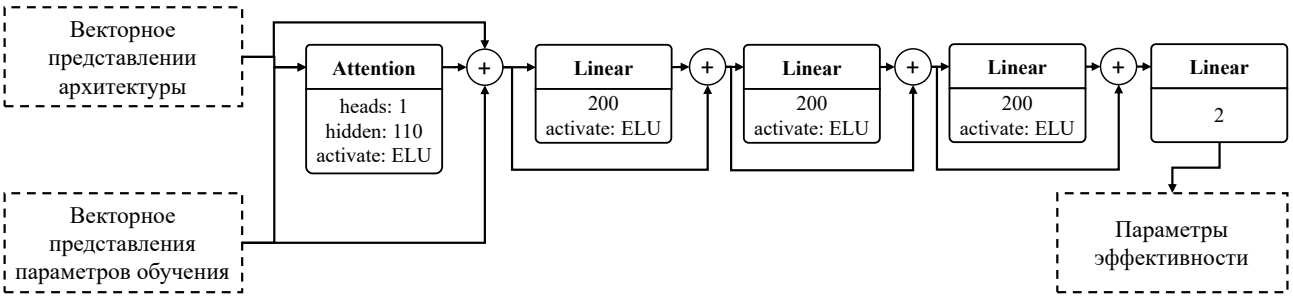

Рис. 4. Формирование прогноза модели

$G \in \mathbb{R}^{10}$, содержащий совокупность числовых признаков двух категорий: гиперпараметры обучения и характеристики вычислительного устройства, на котором осуществляется обучение модели. Список параметров, подаваемых на вход, включает следующие элементы: длина подпоследовательности, количество координат, скорость обучения, объем доступной видеопамяти, пропускная способность памяти, количество тензорных ядер, производительность ускорителя, количество CUDA-ядр, тактовая частота, версия архитектуры (CUDA Capability).

Архитектура Энкодера 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) [25].

3.3. *Формирование прогноза.* На рис. 4 представлена архитектура нейросетевой модели, реализующей 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.

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 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) [26]. Для повышения обобщающей способности модели к полносвязные слоям применяется прореживание (Dropout) [27].

3.4. *Обучение модели.*

3.4.1. Формирование обучающей выборки. Для обучения описанной выше модели используется упорядоченный набор из n четырехэлементных кортежей $M = \{(\widehat{L}_i, R, G, V_i)\}_{i=1}^{n}$, где $\widehat{L}_i$ — матрицы слоев, R_i — матрица связей между слоями, G_i — вектор параметров обучения, V_i — вектор значений качества модели. Каждый элемент $(\widehat{L}_i, R, G, V_i)$ описывает одну 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. Нормализованное значение $v_j \in V_i$ вычисляется на основе следующей формулы:

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

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.

Обучающая выборка представляет собой набор примеров, на которых модель обучается выявлять зависимости между входными и выходными данными. 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 является 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 полагаются следующие данные: кортеж, содержащий подаваемую на вход целевую модель, и вектор параметров качества V_i. Формально обучающая выборка может быть представлена следующий образом:

$$D = \{\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \mid Y_i = (\widehat{L}_i, R_i, V_i), X_i = (\widehat{L}_i, R_i, G_i), 1 \leqslant i \leqslant n\}. \tag{4}$$

Предполагается, что в процессе работы 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^*.

3.4.2. *Вычисление ошибки.* Ошибка $\boldsymbol{E}$ представляет собой составную величину и определяется как взвешенная сумма нескольких компонент, каждая из которых отвечает за

оценку отклонения 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 модели определяется следующим образом:

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

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 модели.

Ошибка прогноза слоя E_{layer} определяется с помощью функции кросс-энтропии [28], вычисляющей расхождение между истинным (one-hot) и предсказанным распределениями вероятностей принадлежности к классам. Согласно формуле (3), 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 расположенные в последних a столбцах $\widehat{L}$ и $\widehat{L}^*$.

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

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 2), тогда как предсказанные были сформированы моделью. Формально ошибка прогноза наличия связей может быть представлена следующим образом:

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

Для вычисления ошибок регрессии используется функция потерь Хубера (Huber loss) [29], которая оценивает расстояние между истинными значениями и предсказанными моделью. В соответствии с формулой (3), 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:

Таблица 1

Аппаратная платформа для экспериментов

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

$$E_{\text{params}} = \frac{1}{\ell}\sum_{i=1}^{\ell} \text{HuberLoss}(\text{slice}_{i,\,c+1:v}(\widehat{L}),\, \text{slice}_{i,\,c+1:v}(\widehat{L}^*)), \quad v = c+2,$$

$$E_{\text{error}} = \frac{1}{\ell}\sum_{i=1}^{c} \text{HuberLoss}(V(i,1),\, V^*(i,1)), \quad E_{\text{time}} = \frac{1}{\ell}\sum_{i=1}^{\ell} \text{HuberLoss}(V(i,2),\, V^*(i,2)),$$

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

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

4. Вычислительные эксперименты. Для исследования эффективности предложенного метода были проведены вычислительные эксперименты, в которых использовалось оборудование Лаборатории суперкомпьютерного моделирования ЮУрГУ [30]. В табл. 1 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.

4.1. *Пространство поиска.* В рамках данного исследования целевой моделью, оптимизируемой в ходе нейросетевого поиска, выступает модель, аппроксимирующая функцию восстановления временного ряда. Целевая модель реализует процесс преобразования неполных подпоследовательностей $\overset{\circ}{\boldsymbol{T}}_{i,\,m}$ в восстановленные $\overset{\bullet}{\boldsymbol{T}}_{i,\,m}$. Обучающая выборка целевой модели формируется из полных подпоследовательностей $\boldsymbol{T}_{i,m}$. В каждую подпоследовательность случайным образом добавляются пропуски до тех пор, пока доля пропущенных точек не превысит 25 % от общего числа элементов. Подпоследовательности с пропусками подаются на вход целевой модели, которая выполняет восстановление. Качество восстановления оценивается путем сравнения выходных восстановленных последовательностей с исходными полными до внесения пропусков.

В результате обучения ожидается, что значения восстановленных подпоследовательностей $\overset{\bullet}{\boldsymbol{T}}_{i,\,m}$ будут приближены к значениям соответствующих полных подпоследовательностей $\boldsymbol{T}_{i,m}$. Для обеспечения объективности оценки качества модели применяется процедура кросс-валидации с несколькими независимыми разбиениями исходных данных на обучающую и тестовую выборки. Оценка точности восстановления осуществляется по тем значениям, которые были искусственно помечены как пропущенные. Для оценки точности восстановления используется среднеквадратичная ошибка (Mean Squared Error, MSE).

Таблица 2

Параметры слоев из пространства поиска

№	Тип слоя	Глубина	Параметры слоя	
			Первый	Второй
1.	Dense	[1,20]	$\{16, 32, 64, 128, 256, 512, 1028\}$	—
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]	$\{16, 32, 64, 128\}$	—

4.1.1. *Параметры целевой модели.* В данном исследовании пространство поиска целевой 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 индивидуальные параметры (см. табл. 2).

Для всех рассматриваемых нейросетевых моделей проводился перебор общих гиперпараметров: длина входной подпоследовательности принимала значения из множества $\{100, 200, 300\}$, скорость обучения из множества $\{0.001, 0.005, 0.0001, 0.01\}$. Для каждой комбинации типов слоев была задана максимально допустимая глубина, учитывающая потенциальный риск исчезновения градиентов при обучении глубоких нейросетей. В совокупности было сформировано дискретное пространство из 200 уникальных моделей, в рамках которого было выполнено более 12 000 запусков обучения.

4.2. *Наборы данных, конкуренты и методика сравнения.* В качестве наборов данных для экспериментов используются результаты обучения моделей, полученные в ходе поиска нейросетевые модели для временных рядов из различных предметных областей. Описание используемых временных рядов представлено в табл. 3. 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 точность прогноза параметров качества каждого исследуемого метода на тестовой выборке с помощью симметричной средней абсолютной процентной ошибке (Symmetric Mean Absolute Percentage Error, SMAPE), которая отражает ошибку в процентах, что облегчает интерпретацию результатов. Формально данная ошибка может быть представлена следующим образом:

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

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

Таблица 3

Наборы данных, используемые в экспериментах

№	Набор	Длина, $n \times 10^3$	Количество, измерений, d	Предметная область
1.	BAFU [31]	50	10	Сброс воды в реках Швейцарии
2.	Electricity [32]	5	9	Потребление электроэнергии в нескольких домашних хозяйствах Франции
3.	Climate [33]	5	10	Погода в различных локациях Северной Америки
4.	Madrid [34]	25	10	Трафик автомобильных дорог в Мадриде
5.	NREL [35]	8.7	9	Потребление электроэнергии в научном центре в США

В качестве конкурентов использовались следующие методы: XGBoost, NGBoost, LightGBM, Random Forest [14], BOHAMIANN [20], DNGO [19], MLP [17], OMNI [18], VSGP [15]. В качестве реализации сравниваемых методов использовалась реализация, предоставляемая в составе фреймворка NASLib [18]. Гиперпараметры конкурентов подбирались индивидуально для каждого временного ряда с использованием платформы Weights & Biases (wandb) [8] в течении одной недели.

4.3. *Результаты.* На рис. 5 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 подразумевается 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 tsGAP2 демонстрирует стабильное и значительное преимущество над конкурентами. В среднем по различным наборам данных tsGAP2 обеспечивает наибольшую точность прогноза как по ошибке модели, так и по времени обучения модели. Величина ошибки прогноза, достигаемая предложенным методом, в среднем составляет 4.4 % по ошибке целеовой 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.

5. Дискуссия. В рамках предлагаемого подхода целевая модель обучается на многомерных временных рядах, которые характеризуются стохастичностью. Значения во временных точках могут формироваться под воздействием множества факторов и подчиняться различным вероятностным закономерностям. Используемая для восстановления целевая модель также имеет стохастический характер. Перед началом обучения веса нейросетевой модели инициализируются случайным образом. В процессе обучения выбор входных и выходных примеров осуществляется случайно. В результате повторное обучение модели может приводить к достижению разных локальных минимумов функции потерь и, соответственно, к вариативности качества модели.

Указанные аспекты свидетельствуют о том, что восстановленные значения и прогноз качества целевой модели не стоит интерпретировать как детерминированные или абсолютно точные оценки. Получаемые результаты целесообразно интерпретировать как ве-

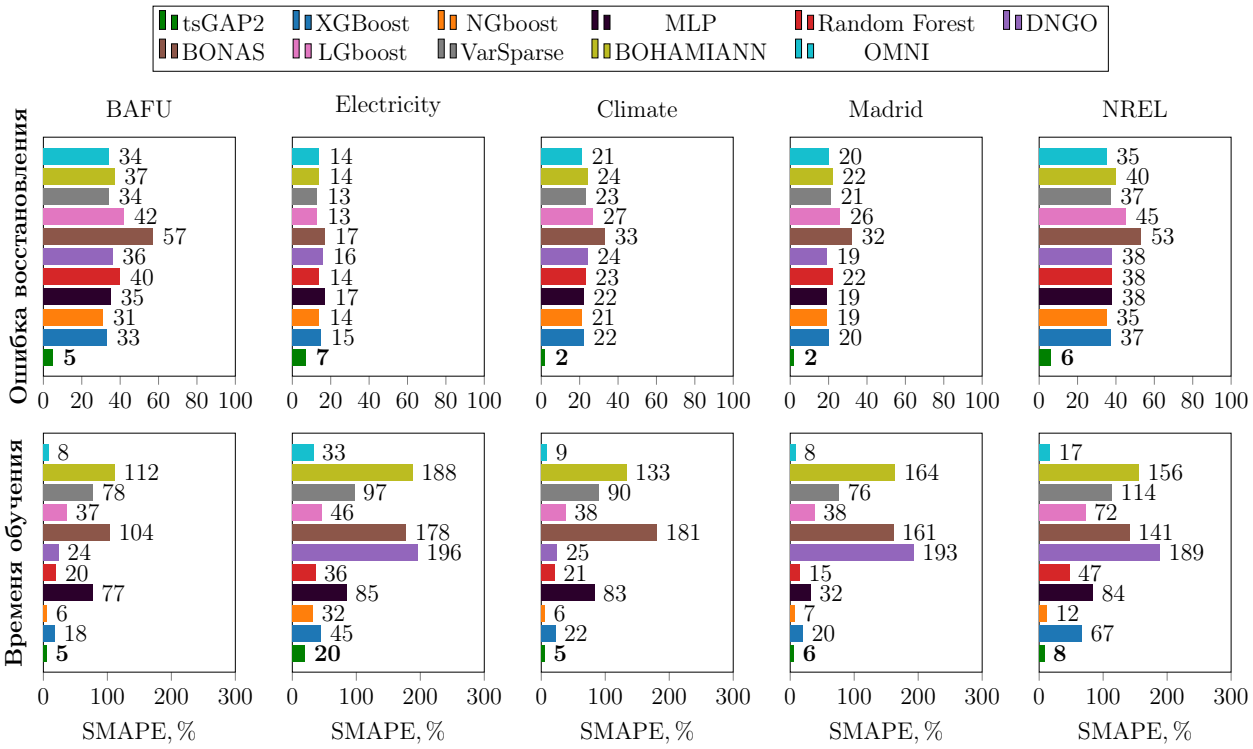

Рис. 5. Результаты сравнения (SMAPE)

роятностные оценки [36], отражающие усредненное поведение модели в рамках обучающей выборки. Точность оценки зависит от репрезентативности обучающей выборки целевой модели и от объема исследованного пространства поиска. Под репрезентативностью в данном контексте следует понимать то, насколько временной ряд отражает характерное разнообразие состояния моделируемой системы.

Важно также отметить, что цель нейросетевой модели заключается не в дословном воспроизведении обучающих данных, а в выявлении и обобщении скрытых закономерностей в наблюдаемых процессах. Несовпадение между восстановленными и исходными значениями необязательно свидетельствует о низкой точности модели. Напротив, чрезмерное соответствие обучающим данным может указывать на переобучение и снижение способности модели к обобщению. Поэтому оценка эффективности должна опираться на ее устойчивость к новым, ранее не встречавшимся входным данным.

Следовательно, несмотря на стохастическую природу как данных, так и моделей, достигнутая точность прогнозов может оставаться в пределах допустимого диапазона, соответствующего требованиям предметной области. Границы допустимых отклонений в этом случае должны определяться либо экспертно, либо исходя из прикладных критериев качества моделирования и анализа. В контексте NAS целью является не абсолютное предсказание качества целевой модели, а обеспечение надежного относительного ранжирования возможных элементов пространства поиска. При таком подходе даже приближенные значения точности оказываются полезными, если сохраняется согласованность в оценке относительного качества альтернатив. Предложенный в данной работе подход демонстрирует сопоставимые или лучшие результаты по сравнению с современными аналогами и,

следуя из изложенного выше, может использоваться для решения задачи прогнозирования качества нейросетевых моделей.

Заключение. В данной статье рассмотрена задача прогноза качества нейросетевых моделей восстановления пропущенных значений в многомерных временных рядах, что является важной проблемой во многих предметных областях. Под качеством нейросетевой модели подразумевается совокупность двух показателей: ошибка модели и время ее обучения на одной эпохе. Предложен метод tsGAP2 для прогнозирования ошибки и времени обучения нейросетевых моделей. В данной работе нейросетевая модель рассматривается как ориентированный ациклический граф, в котором узлы представляют собой слои, а связи представляют собой передачу данных между ними. Метод предполагает наличие трех компонентов: Автоэнкодер графового представления, Энкодер параметров и Агрегатор. Автоэнкодер преобразует графовое представление модели в векторное, содержащее наиболее важную информацию. Энкодер кодирует гиперпараметры и характеристики вычислительного устройства, формируя вектор, содержащий информацию о внешних факторах, влияющих на процесс обучения. Агрегатор объединяет полученные векторы и на их основе прогнозирует показатели качества нейросетевой модели: ошибку на валидационной выборке и время обучения за одну эпоху. Для обучения моделей метода используется составная ошибка, которая представляет собой взвешенную сумму нескольких компонент. Каждая компонента оценивает отклонение по определенному аспекту прогноза: наличию связей между слоями, классификации слоев и функций активации, регрессии параметров слоев, точности и времени работы модели.

Для данного исследования было сформировано и проанализировано пространство поиска, включающее 200 различных нейросетевых моделей. В качестве целевой модели была выбрана нейросетевая модель, выполняющая восстановление временного ряда. Обучение моделей из пространства поиска производилось на временных рядах из различных предметных областей. В ходе исследования пространства поиска было произведено 12 000 запусков обучения моделей. Для оценки эффективности предложенного метода и его сравнения с конкурентами проводились вычислительные эксперименты. В ходе экспериментов модели обучались прогнозировать качество целевой модели. Вычислительные эксперименты показали, что предложенный метод обеспечивает высокую точность предсказания качества целевой модели: средняя ошибка по метрике SMAPE составляет 4.4 %, что существенно превосходит альтернативные подходы, демонстрирующие среднее значение 27.6 %. Средняя ошибка прогноза времени обучения составляет 8.8 %, тогда как существующие методы показывают значительно более высокие значения — до 61.6 %. Дальнейшие исследования будут посвящены разработке методов AutoML, направленных на автоматизацию построения нейростевых моделей восстановления временных рядов и использующий метод tsGAP2 для предсказания качества проектируемой модели.

Список литературы

1. Aydin S. Time series analysis and some applications in medical research // Journal of Mathematics and Statistics Studies. 2022. V. 3. N 2. P. 31–36. DOI: 10.32996/JMSS.

2. Voevodin V. V., Stefanov K. S. Development of a portable software solution for monitoring and analyzing the performance of supercomputer applications // Numerical Methods and Programming. 2023. V. 24. P. 24–36. DOI: 10.26089/NumMet.v24r103.

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

4. Gromov V. A., Lukyanchenko P. P., Beschastnov Yu. N., Tomashchuk K. K. Time Series Structure Analysis of the Number of Law Cases // Proceedings in Cybernetics. 2022. N 4 (48). P. 37–48.

5. Kazijevs M., Samad M. D. Deep imputation of missing values in time series health data: A review with benchmarking // J. Biomed. Informatics. 2023. V. 144. P. 104440. DOI: 10.1016/J.JBI.2023.104440.

6. Elsken T., Metzen J. H., Hutter F. Neural Architecture Search: A Survey // J. Mach. Learn. Res. 2019. V. 20. N 55. P. 1–21. [Electron. res.]: https://jmlr.org/papers/v20/18-598.html.

7. Wozniak A. P., Milczarek M., Wozniak J. MLOps Components, Tools, Process, and Metrics: A Systematic Literature Review // IEEE Access. 2025. V. 13. P. 22166–22175. DOI: 10.1109/ACCESS.2025.3534990.

8. Weights & Biases: Machine learning experiment tracking, dataset versioning, and model management. [El. Res.]: https://wandb.ai/. Access date: 2025-06-11.

9. Bergstra J., Bengio Y. Random search for hyper-parameter optimization // J. Mach. Learn. Res. 2012. V. 13. P. 281–305.

10. Dong X., Yang Y. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search // 8th Int. Conf. on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020. [Electron. res.]: https://openreview.net/forum?id=HJxyZkBKDr.

11. Ding Y., Huang Z., Shou X., Guo Y., Sun Y., Gao J. Architecture-Aware Learning Curve Extrapolation via Graph Ordinary Differential Equation // AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, Feb. 25 — Mar. 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.

12. timeseries Graph Attention Performance Predict. [El. Res.]: https://gitverse.ru/yurtinaa/tsGAP2. Access date: 2025-05-03.

13. Gawlikowski J., Tassi C. R. N., Ali M., Lee J., Humt M., Feng J., Kruspe A., Triebel R., Jung P., Roscher R., Shahzad M., Yang W., Bamler R., Zhu X. X. A survey of uncertainty in deep neural networks // Artif. Intell. Rev. 2023. V. 56. N 1. P. 1513–1589. ISSN: 1573–7462. DOI: 10.1007/s10462-023-10562-9.

14. Zela A., Siems J. N., Zimmer L., Lukasik J., Keuper M., Hutter F. Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks // The Tenth Int. Conf. on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022. [Electron. res.]: https://openreview.net/forum?id=0npFa95RVqs.

15. Titsias M. Variational Learning of Inducing Variables in Sparse Gaussian Processes // Proc. of the Twelfth Int. Conf. on Artificial Intelligence and Statistics. / ed. by D. van Dyk, M. Welling. Hilton Clearwater Beach Resort, Clearwater Beach, Florida, USA: PMLR, 16–18 Apr. 2009. V. 5. P. 567–574. [Electron. res.]: https://proceedings.mlr.press/v5/titsias09a.html.

16. Ying C., Klein A., Christiansen E., Real E., Murphy K., Hutter F. NAS-Bench-101: Towards Reproducible Neural Architecture Search // Proc. of the 36th Int. Conf. on Machine Learning, ICML 2019, June 9–15, Long Beach, California, USA / ed. by K. Chaudhuri, R. Salakhutdinov. PMLR, 2019. V. 97. P. 7105–7114. [Electron. res.]: http://proceedings.mlr.press/v97/ying19a.html.

17. White C., Neiswanger W., Savani Y. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search // Thirty-Fifth AAAI Conf. on Artificial Intelligence, AAAI 2021, IAAI 2021, EAAI 2021, Virtual Event, Feb. 2–9, 2021. AAAI Press, 2021. P. 10293–10301. DOI: 10.1609/AAAI.V35I12.17233.

18. White C., Zela A., Ru R., Liu Y., Hutter F. How powerful are performance predictors in neural architecture search? // Adv. Neural Inf. Process. Syst. 2021. V. 34. P. 28454–28469.

19. Snoek J., Rippel O., Swersky K., Kiros R., Satish N., Sundaram N., Patwary M., Prabhat, Adams R. P. Scalable Bayesian Optimization Using Deep Neural Networks // Proc. of the 32nd Int. Conf. on Machine Learning (ICML). Lille, France: PMLR, 2015. V. 37. P. 2171–2180.

20. Springenberg J. T., Klein A., Falkner S., Hutter F. Bayesian Optimization with Robust Bayesian Neural Networks // Adv. Neural Inf. Process. Syst / ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett. V. 29.

21. Wu X., Zhang D., Guo C., He C., Yang B., Jensen C. S. AutoCTS: Automated Correlated Time Series Forecasting // Proc. VLDB Endow. 2021. V. 15. N 4. P. 971–983. DOI: 10.14778/3503585.3503604.

22. Wang C., Chen X., Wu C., Wang H. AutoTS: Automatic Time Series Forecasting Model Design Based on Two-Stage Pruning // arXiv preprint: abs/2203.14169. DOI: 10.48550/arXiv.2203.14169.

23. Velickovic P., Cucurull G., Casanova A., Romero A., Li'o P., Bengio Y. Graph Attention Networks // 6th Int. Conf. on Learning Representations, ICLR 2018, Vancouver, Canada, April 30 — May 3, 2018. 2018. [Electron. res.]: https://openreview.net/forum?id=rJXMpikCZ.

24. Clevert D., Unterthiner T., Hochreiter S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) // 4th Int. Conf. on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016 / ed. by Y. Bengio, Y. LeCun. 2016. [Electron. res.]: http://arxiv.org/abs/1511.07289.

25. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. V. 6. N 2. P. 107–116. DOI: 10.1142/S0218488598000094.

26. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // 2016 IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, USA. IEEE Computer Society. 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.

27. Srivastava N., Hinton G. E., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting // J. Mach. Learn. Res. 2014. V. 15. N 1. P. 1929–1958. DOI: 10.5555/2627435.2670313.

28. Mao A., Mohri M., Zhong Y. Cross-Entropy Loss Functions: Theoretical Analysis and Applications // Proc. of the 40th Int. Conf. on Machine Learning / ed. by A. Krause. 2023. V. 202. P. 23803–23828.

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

30. Bilenko R. V., Dolganina N. Yu., Ivanova E. V., Rekachinsky A. I. High-performance Computing Resources of South Ural State University // Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. V. 11. N 1. P. 15–30. DOI: 10.14529/cmse220102.

31. BundesAmt Für Umwelt — Swiss Federal Office for the Environment. [El. Res.]: https://www.hydrodaten.admin.ch/. Access date: 2025-05-03.

32. Trindade A., "Electricity Load Diagrams 2011–2014," UCI Machine Learning Repository (2015) [El. Res.]: https://doi.org/10.24432/C58C86. Access date: 2023-05-03.

33. Lozano A. C., Li H., Niculescu-Mizil A., Liu Y., Perlich C., Hosking J. R. M., Abe N. Spatial-temporal causal modeling for climate change attribution // Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Paris, France, June 28 — July 1, 2009 / ed. by J. F. Elder IV, F. Fogelman-Soulié, P. A. Flach, M. J. Zaki. — ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

34. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques // Transp. Res. Part C: Emerg. Technol. 2018. V. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

35. Sheppy M., Beach A., Pless S. NREL RSF Measured Data 2011. [El. Res.]: `https://data.openei.org/submissions/358`. Access date: 2023-09-03.

36. Snytnikov A. V., Ezrokh Yu. S. Solving Vlasov Equation with Neural Networks // Lobachevskii Journal of Mathematics. 2024. V. 45. P. 3416–3423.

Юртин Алексей Артемьевич — программист Лаборатории больших данных и машинного обучения, аспирант кафедры системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет), победитель конкурсного отбора 2025 года на назначение стипендии Президента Российской Федерации для аспирантов и адъюнктов. Сфера научных интересов: машинное обучения, обработка временных рядов, искусственный интеллект. E-mail: iurtinaa@susu.ru.

Yurtyn Aleksei Artemyevich — Programmer at the Laboratory of Big Data and Machine Learning, PhD student at the Department of System Programming, South Ural State University (National Research University); winner of the 2025 competitive selection for the Presidential Scholarship of the Russian Federation for postgraduate students and adjuncts. Research interests: machine learning, time series analysis, artificial intelligence. E-mail: iurtinaa@susu.ru.

Дата поступления — 04.07.2025