УДК 004.82

ИСПОЛЬЗОВАНИЕ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ И ВЫЯВЛЕНИЯ ТЕНДЕНЦИЙ В ДАННЫХ

Чуб Вадим Сергеевич[1],
аспирант,
e-mail: vadim-chub13@mail.ru,
[1]*Донской государственный технический университет, г. Ростов-на-Дону, Россия*

Традиционные модели рекуррентной нейронной сети RNN (Recurrent Neural Network), такие как LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit), давно применяются для анализа временных рядов, но обладают ограничениями в обучении на длинных последовательностях и высокими вычислительными требованиями. Последние достижения в области прогнозирования временных рядов показали снижение их значимости. В данной работе предложена новая архитектура RNN с линейной временной сложностью и меньшими затратами памяти. В ходе исследования проведена апробация предложенной модели RNN и оценка ее производительности на различных задачах анализа временных рядов. Представлены результаты эксперимента, которые показывают, что предложенная модель превосходит современные альтернативы. Конкурентоспособность модели была подтверждена сравнением с передовыми моделями, такими как PatchTST и TimesNet. Кроме того, предложенная модель превзошла по производительности модели на основе многослойных перцептронов (MLP) и оказалась более эффективной, чем модели на основе трансформеров. Предложенная архитектура RNN может стать перспективным направлением для будущих исследований в данной области.

Ключевые слова: рекуррентные нейронные сети, временные ряды, трансформеры, долгосрочные зависимости, энкодер

THE USE OF RECURRENT NEURAL NETWORKS TO ANALYZE TIME SERIES AND IDENTIFY TRENDS IN DATA

Chub V.S.[1],
post-graduate student,
e-mail: vadim-chub13@mail.ru,
[1]*Don State Technical University, Rostov-on-Don, Russia*

Traditional models of the recurrent neural network (RNN), such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit), have long been used for time series analysis, but have limitations in learning on long sequences and high computational requirements. Recent advances in time series forecasting have shown a decrease in their importance. In this article, a new RNN architecture with linear temporal complexity and lower memory consumption is proposed. During the study, the proposed RNN model was tested and its performance was evaluated for various temporal series analysis tasks. The presented results of the experiment show that the proposed model is superior to modern alternatives. The competitiveness of the model was confirmed by comparison with advanced models such as PatchTST and TimesNet. In addition, the proposed model surpassed the performance of models based on multilayer perceptrons (MLP) and proved to be more effective than models based on transformers. The proposed RNN architecture may become a promising direction for future research in this field.

Keywords: recurrent neural networks, temporal series, transformers, long-term dependencies, encoder

DOI 10.21777/2500-2112-2024-2-91-102

Введение

Анализ временных рядов является фундаментальной областью в машинном обучении с различными приложениями, от финансового прогнозирования до прогнозирования окружающей среды. Исторически традиционные рекуррентные нейронные сети (RNN) были основополагающим инструментом для моделирования последовательных данных благодаря их способности захватывать временные зависимости. В последнее время традиционные RNN постепенно утратили свое доминирующее положение в задачах временных рядов – большей популярностью пользуются альтернативные архитектуры, такие как трансформеры, многослойные персептроны (MLP) и сверточные нейронные сети (CNN).

Традиционные рекуррентные нейронные сети (RNN), в своей базовой форме или в виде вариантов, таких как Long Short-Term Memory (LSTM) и Gated Recurrent Unit (GRU), обладают следующими ограничениями [1]:

– проблема затухающего/взрывного градиента ограничивает возможность захвата информации в длинных последовательностях. При длине последовательности более ста единиц способность к захвату информации быстро снижается;

– невозможность проведения параллельных вычислений приводит к низкой вычислительной эффективности и затрудняет возможность масштабирования;

– подход к прогнозированию «шаг за шагом» приводит к накоплению ошибок и низким скоростям вывода.

Последние исследования утверждают, что RNN больше не являются оптимальным выбором для задач временных рядов, связанных с моделированием долгосрочных зависимостей [2–4]. Для того, чтобы преодолеть ограничения традиционных RNN, в данном исследовании ставится задача создания модели на основе RNN для задач временных рядов, отличающаяся тремя основными особенностями:

1) новая архитектура RNN, характеризующаяся линейной временной сложностью и меньшими затратами памяти;

2) улучшенная способность захватывать информацию о долгосрочных последовательностях, превосходящая возможности традиционных RNN;

3) высокая вычислительная эффективность с возможностью масштабирования.

Основная часть

Современные исследования [2–4] свидетельствуют о возрастающей популярности моделей трансформеров в задачах временных рядов благодаря их возможностям параллелизации и механизмам внимания, обеспечивающим эффективную обработку последовательных данных. Аналогично, CNN продемонстрировали силу в извлечении локальных паттернов и признаков из временных данных, показав перспективные результаты в различных научных областях. Модели на основе RNN долгое время были предпочтительным выбором для задач прогнозирования временных рядов из-за их способности обрабатывать последовательные данные. В последнее время проделана большая работа по применению RNN для краткосрочного и вероятностного прогнозирования, что привело к значительным достижениям [5; 6]. Однако в области прогнозирования долгосрочных последовательностей с расширенными историческими данными и горизонтами прогнозирования RNN считаются неэффективными в выявлении долгосрочных зависимостей, что приводит к постепенному отказу от них. С другой стороны, новые архитектуры RNN добились значительного успеха в области крупномасштабных языковых моделей [7; 8], продемонстрировав производительность, сравнимую с моделями трансформеров, обладая при этом высокой эффективностью.

В последние годы было проделано значительное количество исследований, пытающихся применить модели трансформеров к прогнозированию долгосрочных временных рядов. Здесь мы подводим краткий итог некоторых из этих работ. Модель в работе [9] использует слои сверточного само-внимания для захвата локальной информации и снижения пространственной сложности. Еще одна модель вводит механизм само-внимания для эффективного извлечения наиболее важной информации [2]. Существует

модель, которая использует улучшенные структуры Фурье для достижения линейной сложности [4]. Кроме того, стоит упомянуть об использовании предварительно обученных моделей GPT для задач временных рядов.

Многослойные персептроны (MLP) широко применяются в прогнозировании временных рядов [10]. Авторы работы [11] опередили современные модели на основе трансформеров, добавив линейный слой и стратегию канальной обработки. Этот успех поспособствовал разработке многочисленных моделей на основе MLP [12; 13]. Достижения моделей на основе MLP вызвали вопросы о необходимости применения сложных преобразователей для прогнозирования временных рядов.

Последние исследования также свидетельствуют о возрастающей роли моделей CNN в области временных рядов [14].

Предложенная модель, архитектура которой изображена на рисунке 1, использует энкодер в качестве основного модуля обработки. Модуль ввода применяет нормализацию экземпляра к одномерному ряду каждого канала и сегментирует их на патчи, которые служат входными токенами сети. Они поступают на вход оператора, который включает модули временного микширования и микширования каналов. Выходные данные последнего слоя выравниваются и используются для прогнозирования цели. Таким образом, архитектура предложенной модели состоит из трех основных компонентов: модуля ввода, многоголового оператора[1] и модуля вывода.

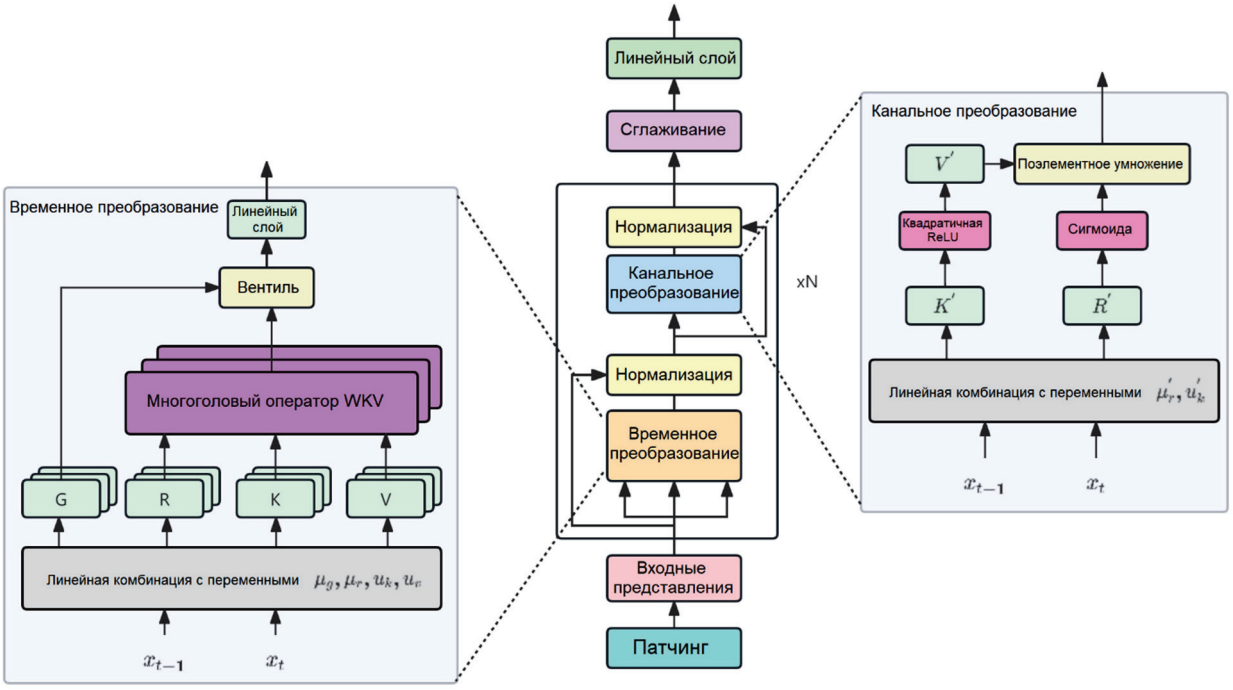

Рисунок 1 – Механизм работы модели

Модуль ввода применяет нормализацию экземпляра к одномерному ряду каждого канала и сегментирует их на патчи, которые служат входными токенами сети. Он нормализует каждый одномерный временной ряд с нулевым средним и единичным стандартным отклонением, смягчая эффект сдвига распределения между обучающими и тестовыми данными. Одномерный временной ряд разбивается на патчи, уменьшая количество входных токенов, что помогает модели использовать меньше вычислений и памяти.

Многоголовый оператор включает модули временного микширования и микширования каналов, выполняет временное преобразование и само-внимание с линейной сложностью по времени и про-

[1] В данном контексте термин «многоголовый» (*multihead*) относится к механизму многоголового внимания (*multi-head attention*), который используется в различных архитектурах нейронных сетей, включая трансформеры и модифицированные рекуррентные нейронные сети (RNN).

странству. Механизм временного смешивания использует линейные комбинации текущих и предыдущих временных шагов для интерполяции между входами, обеспечивая эффективную передачу информации. Механизм само-внимания (WKV[2]) вводит многоголовость для увеличения емкости модели, с операциями временно-зависимого обновления векторов и линейной комбинацией для эффективного извлечения информации.

Модуль вывода выравнивает выходные данные и использует их для прогнозирования цели. Вектор вывода реализуется с использованием функции активации SiLU и приемника, где каждый временной шаг нормализуется отдельно, обеспечивая точность прогнозирования.

Нормализация и патчинг

В работе рассматривается задача анализа многомерных временных рядов. Пусть дан набор данных, содержащий несколько многомерных временных рядов $x \in \mathbb{R}^{L \times M}$, где L – длина входной последовательности и M – количество одномерных временных рядов. Каждый одномерный временной ряд $x_i \in \mathbb{R}^L$ представляет собой отдельный экземпляр, где i обозначает индекс одного из M временных рядов.

Первый шаг – это нормализация экземпляра. Эта техника была недавно предложена для смягчения эффекта сдвига распределения между обучающими и тестовыми данными [15]. Для каждого одномерного временного ряда экземпляр нормализуется с нулевым средним и единичным стандартным отклонением. По сути, каждый экземпляр нормализуется перед формированием патчей, а затем среднее и отклонение возвращаются к выходному прогнозу.

На втором шаге каждый входной одномерный временной ряд x_i токенизируется путем формирования патчей. Одномерный временной ряд разбивается на L патчей, которые могут перекрываться. Если обозначить длину патча как P, а шаг (неперекрывающуюся область между двумя последовательными патчами) как S, то процесс формирования патчей сгенерирует последовательность патчей $x_i^p \in R^{N \times P}$, где N – количество патчей, которое можно вычислить следующим образом:

$$N = \frac{(L - P)}{S} + 2. \quad (1)$$

С помощью формирования патчей количество входных токенов было уменьшено с L до N согласно (1). Это уменьшение помогает модели использовать меньше вычислений и меньше памяти. Наконец, последовательность патчей $x_i^p \in R^{N \times P}$ проецируется на входные токены $x_i^t \in R^{N \times D}$ через обучаемую матрицу проекции, где D – размерность скрытого пространства, в которую проецируются патчи после их формирования. Проекция позволяет модели преобразовывать последовательность патчей в формат, удобный для дальнейшей обработки и анализа.

Основа сети структурирована с использованием сложенных блоков остатков, каждый из которых содержит подблок временного смешивания и подблок канального смешивания. Эти компоненты воплощают в себе рекуррентные структуры, разработанные для использования ранее изученной информации. Сеть работает в параллельном и рекуррентном режиме. Параллельный режим более выгоден для параллельного обучения и обеспечивает более высокую вычислительную эффективность, служа основным режимом для реализации кода. Основные формулы для параллельного режима представлены ниже.

Блок временного преобразования

В модели используются обучаемые переменные $\mu_g, \mu_r, \mu_k, \mu_v$ в линейной комбинации x_t и x_{t-1}, чтобы достичь простого временного смешивания, которое состоит в интерполяции между входами текущего и предыдущего временных шагов. Комбинация сдвинутого предыдущего шага и текущего шага была линейно спроектирована через матрицу проекции внутри блока:

[2] Разработано автором. В контексте предложенной модели RNN аббревиатура “WKV” относится к компонентам оператора само-внимания, используемого для временного и канального микширования входных данных. Здесь **W** – веса (*Weights*), используемые в линейной комбинации; **K** – ключи (*Keys*), представляющие собой проекции входных данных; **V** – значения (*Values*), представляющие собой другие проекции входных данных, которые комбинируются с ключами для создания выходных значений.

$$g_t = W_g \cdot \left(\mu_g \odot x_t + \left(1-\mu_g\right) \odot x_{t-1}\right), \quad (2)$$

$$r_t = W_r \cdot \left(\mu_r \odot x_t + \left(1-\mu_r\right) \odot x_{t-1}\right), \quad (3)$$

$$k_t = W_k \cdot \left(\mu_k \odot x_t + \left(1-\mu_k\right) \odot x_{t-1}\right), \quad (4)$$

$$v_t = W_v \cdot \left(\mu_v \odot x_t + \left(1-\mu_v\right) \odot x_{t-1}\right). \quad (5)$$

В формулах (2)–(5) приняты следующие обозначения:

W_g, W_r, W_k, W_v – обучаемые матрицы проекций, соответствующие различным компонентам временного смешивания. Эти матрицы преобразуют входные данные в новое пространство признаков:

W_g – матрица проекции для компоненты g;

W_r – матрица проекции для компоненты r;

W_k – матрица проекции для компоненты k;

W_v – матрица проекции для компоненты v;

“$\cdot$” – матричное умножение, выполняемое между матрицами или матрицами и векторами;

$\odot$ – операция поэлементного умножения (Hadamard product), которая применяется к соответствующим элементам двух векторов или матриц.

Сдвиг токенов – простой и интуитивно понятный способ временного преобразования, обеспечивающий эффективную передачу информации с точки зрения механизма. Более того, все эти операции являются линейными, что позволяет легко выполнять параллельные вычисления.

В сети многоголовый оператор WKV отражает само-внимание, но с линейной сложностью по времени и пространству. Это повторяющееся поведение в архитектуре проявляется через временно-зависимое обновление векторов WKV. Формула оператора WKV для одной головы выглядит следующим образом:

$$wkv_t = \operatorname{diag}\left(u\right) \cdot k_t^{\mathrm{T}} \cdot v_t + \sum_{i=1}^{t-1} \operatorname{diag}(w)^{t-1-i} \cdot k_i^{\mathrm{T}} \cdot v_i, \quad (6)$$

где w и u – два обучаемых параметра;

$\operatorname{diag}\left(u\right)$ – диагональная матрица, построенная из вектора u. Диагональная матрица содержит элементы вектора u на главной диагонали, а остальные элементы равны нулю. Обозначение $\operatorname{diag}\left(u\right)$ указывает на преобразование вектора u в диагональную матрицу;

k_t^{T} – транспонированная матрица ключей для текущего временного шага t. Здесь k_t представляет собой вектор ключей для временного шага t, а транспонирование k_t^{T} позволяет выполнить матричное умножение;

$\operatorname{diag}(w)$ – диагональная матрица, построенная из вектора w. Как и в случае с $\operatorname{diag}\left(u\right)$, $\operatorname{diag}\left(w\right)$ преобразует вектор w в диагональную матрицу.

Параметр u – это бонус, который награждает модель за встречу токена в первый раз, конкретно текущего токена. Это помогает модели уделять больше внимания текущему токену и обходить любое потенциальное ухудшение w. Еще одним важным параметром в (6) является сам w – вектор канальной временной декомпозиции каждой головы. Здесь термин «голова» относится к одной из независимых проекций входных данных в механизме многоголового внимания. Каждая голова обрабатывает данные параллельно и независимо, что позволяет модели извлекать различные аспекты информации из входной последовательности.

Кроме того, параметр w преобразуется следующим образом:

$$w = \exp\left(-\exp\left(w\right)\right). \quad (7)$$

Преобразование (7) гарантирует, что все значения w находятся в диапазоне (0, 1), обеспечивая, что diag(w) представляет собой матрицу сжатия.

В отличие от одной головы, используемой в обычной сети RWKV [7], предложенная модель вводит механизм многоголовости для увеличения емкости модели. Многоголовый (multihead) оператор WKV формально описан следующим уравнением:

$$\text{multihead } wkv_t = \text{Объединение}\left(wkv_t^1, \ldots, wkv_t^h\right), \tag{8}$$

где h – количество голов.

Однако на практике не выполняется прямое объединение (8). Вместо этого используются операции изменения формы входных данных (векторов ключей k, значений v и весов w) для параллельных вычислений, что позволяет осуществлять параллельные вычисления для каждой головы в механизме многоголового внимания, а затем форма результата многоголового внимания после параллельных вычислений для каждой головы меняется до эквивалентной исходной формы входных данных перед изменением формы. Это необходимо для того, чтобы результат многоголового внимания мог быть корректно использован на следующих этапах обработки данных модели.

Выходной слой (output gate) реализуется с использованием функции активации SiLU и приемника. Вектор вывода на каждую голову задается в виде (9):

$$o_t = \left(\text{SiLU}\left(g_t\right) \odot \text{LayerNorm}\left(r_t \cdot wkv_t\right)\right) W_o, \tag{9}$$

где слой нормализации LayerNorm действует на каждую из h голов отдельно, что также эквивалентно операции групповой нормализации на h группах [16].

Блок канального преобразования

В этом блоке каналы смешиваются сильными нелинейными операциями следующим образом:

$$k_t^{'} = W_g^{'} \cdot \left(\mu_k^{'} \odot x_t + \left(1 - \mu_k^{'}\right) \odot x_{t-1}\right), \tag{10}$$

$$r_t^{'} = W_r^{'} \cdot \left(\mu_r^{'} \odot x_t + \left(1 - \mu_r^{'}\right) \odot x_{t-1}\right), \tag{11}$$

$$v_t^{'} = ReLU^2\left(k_t^{'}\right) \cdot W_v^{'}, \tag{12}$$

$$o_t^{'} = \text{Sigmoid}\left(r_t^{'}\right) \odot v_t^{'}. \tag{13}$$

В формулах (10)–(13) используются следующие обозначения:

$k_t^{'}$ – промежуточный вектор ключей после преобразования в блоке канального преобразования;

$W_g^{'}$ – обучаемая матрица проекции для компонента $k_t^{'}$;

$\mu_k^{'}$ – обучаемый вектор, используемый для временного смешивания входных данных x_t и x_{t-1};

$r_t^{'}$ – промежуточный вектор после применения матрицы проекции $W_r^{'}$ в блоке канального преобразования;

$W_r^{'}$ – обучаемая матрица проекции для компонента $r_t^{'}$;

$\mu_r^{'}$ – обучаемый вектор, используемый для временного смешивания входных данных x_t и x_{t-1};

$v_t^{'}$ – промежуточный вектор значений после применения нелинейной функции активации ReLU и умножения на матрицу проекции $W_v^{'}$;

ReLU^2 – квадрат функции активации ReLU. Применение ReLU^2 к $k_t^{'}$ позволяет получить $v_t^{'}$;

$W_v^{'}$ – обучаемая матрица проекции для компоненты $v_t^{'}$;

$o_t^{'}$ – выходной вектор после применения сигмоидной функции активации к $r_t^{'}$ и поэлементного умножения с $v_t^{'}$;

Sigmoid – сигмоидальная функция активации, которая нормализует значения $r_t^{'}$ в диапазоне от 0 до 1;

$\odot$ – операция поэлементного умножения (Yadamard product).

Рекуррентный режим

Значительной особенностью сети RWKV является тот факт, что у этой сети есть одна модель с двумя наборами формул, позволяющая параллельному режиму быть записанным как рекуррентный

режим, и эти два режима полностью эквивалентны. Термин WKV можно альтернативно записать в рекуррентном режиме (14), (15):

$$wkv_t = s_{t-1} + \mathrm{diag}(u) \cdot k_t^{\mathrm{T}} \cdot v_t, \tag{14}$$

$$s_t = \mathrm{diag}(w) \cdot s_{t-1} + k_t^{\mathrm{T}} \cdot v_t, \tag{15}$$

где s_t – состояние текущего временного шага;

s_{t-1} – состояние предыдущего временного шага.

Функция потерь

Функция потерь, используемая для модели, представляет среднеквадратичную ошибку (MSE), которая определяется выражением

$$\mathrm{MSE}\left(y, \hat{y}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2, \tag{16}$$

где y_i – фактическое значение;

$\hat{y}_i$ – прогнозируемое значение.

Экспериментальная часть

В данном разделе проводится оценка и сравнительный анализ показателей предложенной модели рекуррентных нейронных сетей (RNN) с линейной временной сложностью и оптимизированным использованием памяти.

В таблице 1 приведено сравнение различных современных архитектур с предложенной в работе моделью.

Таблица 1 – Сравнение сложности моделей по затратам времени и памяти[3]

Модель	Время	Память	Шаги тестирования
Предложенная модель	O(L)	O(L)	1
LSTM	O(L)	O(L)	L
DLinear	O(L)	O(L)	1
TimesNet	$O(k^2L)$	O(L)	1
Трансформер	$O(L^2)$	$O(L^2)$	L
Informer	O(L log L)	O(L log L)	1
Autoformer	O(L log L)	O(L log L)	1
LogTrans	O(L log L)	$O(L^2)$	1
PatchTST	$O(L^2)$	$O(L^2)$	1

Здесь L – длина последовательности, k – размер ядра сверток. Модели классифицированы по четырем типам на основе их архитектуры: RNN, MLP, CNN и трансформер. Поскольку предложенная модель является линейной RNN-моделью, ее временная и пространственная сложности составляют O(L), что превосходит сложность $O(L^2)$ обычного трансформера и сравнимо с самой эффективной моделью DLinear. Кроме того, поскольку модель использует архитектуру энкодера, ее тестирование выполняется за один шаг. Это обеспечивает эффективность и избегает проблемы накопления ошибок, наблюдаемой в традиционных моделях RNN.

По сравнению с традиционными RNN предложенная модель имеет следующие существенные преимущества:

1. Параллельные вычисления. Модель использует линейную конструкцию RNN, что позволяет осуществлять параллельные вычисления, и это способствует высокой вычислительной эффективности и возможности масштабирования. В этом состоит отличие от LSTM/GRU, которые нельзя параллельно

[3] Разработано автором.

выполнить из-за их нелинейной зависимости от последнего скрытого состояния, что приводит к более низкой эффективности и неспособности масштабирования.

2. Увеличенная способность захвата информации на длинные расстояния. Традиционные RNN, такие как LSTM и GRU, сталкиваются с заметным узким местом, когда длина последовательности превышает 100, и потери токенов не могут быть дальше снижены; в то время как предложенная модель продемонстрировала способность к эффективному захвату информации, даже когда длина последовательности достигает 4096, с непрерывным снижением потерь токенов. Это в значительной степени связано с хорошо спроектированными механизмами сдвига токенов и временного затухания, которые облегчают передачу информации внутри модели.

3. Архитектура на основе энкодера. Модель интегрирует самые передовые идеи в области временных рядов, используя архитектуру энкодера для обработки информации о временных рядах, отказавшись от рекуррентной итерационной структуры традиционной RNN. Это позволяет иметь только один шаг вывода и устраняет накопление ошибок.

Обширные эксперименты на основных типах задач, связанных с последующим этапом обработки данных, подтверждают производительность и эффективность предложенной модели. Среди таких задач классификация временных рядов, обнаружение аномалий, восполнение пропущенных значений, краткосрочное и долгосрочное прогнозирование, а также прогнозирование по небольшому числу данных.

Прежде всего, у предложенной модели есть наиболее прямое преимущество в эффективности перед PatchTST (модель на основе трансформатора) и TimesNet (модель на основе CNN) из-за ее временной и пространственной сложности O(L). Анализ вычислительных затрат является ключевым для изучения практичности модели.

Результаты эксперимента представлены в таблице 2. Время обучения измеряется для каждого шага, а время вывода измеряется для каждой партии. Каждая базовая и предложенная модели настроены с размерностью скрытого слоя 768 и состоят из трех слоев. Как следует из результатов, показатели модели состоят в значительном улучшении временной эффективности и снижении количества параметров по сравнению с базовыми моделями с аналогичными размерами. Каждая из моделей FEDformer, TimesNet и PatchTST является передовой в той или иной области.

Таблица 2 – Сравнение затрат на обучение и вывод[4]

Модель	Число параметров	Время обучения, с	Время вывода, с
Предложенная модель	24M	0.067	0.018
FEDFormer	33M	0.208	0.056
TimesNET	42M	5.723	2.162
PatchTST	20M	0.457	0.123

Предложенная модель демонстрирует конкурентоспособную производительность по сравнению с современными моделями в различных областях, но заметно превосходит их в плане эффективности.

Все результаты усреднены по четырем различным длинам предсказания, то есть {24, 36, 48, 60} для модели ILI и {96, 192, 336, 720} для остальных. Для долгосрочного прогнозирования эксперимент проводился следующим образом: было использовано восемь хорошо известных наборов реальных эталонных данных, в том числе данные о погоде, дорожном движении, электричестве, болезнях и четыре набора данных ETT (ETTh1, ETTh2, ETTm1, ETTm2) для оценки долгосрочного прогнозирования эффективности.

Экспериментальные результаты сравнения долгосрочного прогнозирования представлены в таблице 3. Предложенная модель демонстрирует уровень производительности, сравнимый с PatchTST, и превосходит другие базовые модели. Особенно заметно ее превосходство по сравнению с недавним методом TimesNet. Модель демонстрирует относительное снижение среднеквадратической ошибки на 12,58 % и средней абсолютной ошибки на 4,38 %. Производительность предложенной LSTM-модели ниже на 4,01 % при разбиении по пять тысяч экземпляров с ухудшением на 5,26 %. В разбиении до одной тысячи точность падает на 8,6 %, в то время как деградация составляет 11,28 %.

[4] Разработано автором.

Таблица 3 – Задача долгосрочного прогнозирования[5]

Методы/ Модели	Новая модель MSE	Новая модель MAE	TimesNet MSE	TimesNet MAE	ETSformer MSE	ETSformer MAE	LightTS MSE	LightTS MAE	DLinear MSE	DLinear MAE
Weather	0.231	0.266	0.259	0.287	0.271	0.334	0.261	0.312	0.249	0.3
ETTh1	0.433	0.445	0.458	0.45	0.542	0.51	0.491	0.479	0.423	0.437
ETTh2	0.375	0.412	0.414	0.427	0.439	0.452	0.602	0.543	0.431	0.447
ETTm1	0.376	0.401	0.4	0.406	0.429	0.425	0.435	0.437	0.357	0.378
ETTm2	0.287	0.338	0.291	0.333	0.293	0.342	0.409	0.436	0.267	0.334
ILI	1.91	0.925	2.139	0.931	2.497	1.004	7.382	2.003	2.169	1.041
ECL	0.159	0.253	0.192	0.295	0.208	0.323	0.229	0.329	0.166	0.263
Traffic	0.398	0.276	0.62	0.336	0.621	0.396	0.622	0.392	0.434	0.295
Среднее	0.521	0.414	0.596	0.433	0.662	0.473	1.303	0.616	0.562	0.436
Методы/ Модели	**FEDformer MSE**	**FEDformer MAE**	**PatchTST MSE**	**PatchTST MAE**	**Stationary MSE**	**Stationary MAE**	**Autoformer MSE**	**Autoformer MAE**	**Informer MSE**	**Informer MAE**
Weather	0.309	0.36	0.225	0.264	0.288	0.314	0.338	0.382	0.634	0.548
ETTh1	0.44	0.46	0.413	0.43	0.57	0.537	0.496	0.487	1.04	0.795
ETTh2	0.437	0.449	0.33	0.379	0.526	0.516	0.45	0.459	4.431	1.729
ETTm1	0.448	0.452	0.351	0.387	0.481	0.456	0.588	0.517	0.961	0.734
ETTm2	0.305	0.349	0.255	0.315	0.306	0.347	0.327	0.371	1.41	0.81
ILI	2.847	1.144	1.443	0.798	2.077	0.914	3.006	1.161	5.137	1.544
ECL	0.214	0.327	0.161	0.253	0.193	0.296	0.227	0.338	0.311	0.397
Traffic	0.61	0.376	0.39	0.264	0.624	0.34	0.628	0.379	0.764	0.416
Среднее	0.701	0.489	0.446	0.386	0.633	0.465	0.757	0.511	1.836	0.871

Для комплексной оценки разнообразных алгоритмов в задачах прогнозирования были дополнительно проведены эксперименты по краткосрочному прогнозированию с использованием набора данных M4, который включает в себя маркетинговые данные различной частоты.

Результаты, представленные в таблице 4, показывают, что предложенная модель превзошла несколько моделей на основе трансформаторов, таких как ETSformer, FEDformer, Informer, Autoformer и Stationary Transformer. Кроме того, результаты также указывают на то, что по сравнению с моделями на базе MLP, такими как DLinear и LightTS, модель демонстрирует лучшую производительность. Разница между предложенной моделью и современными моделями, такими как TimesNet и N-BEATS, также незначительна. Модель RNN продемонстрировала конкурентоспособную производительность на различных задачах анализа временных рядов, включая долгосрочное и краткосрочное прогнозирование, обнаружение аномалий, классификацию и работу с небольшими объемами данных.

Таблица 4 – Задача краткосрочного прогнозирования[6]

Методы/ Модели	Новая модель	TimesNet	PatchTST	N-Hits	N-BEATS	ETSformer
SMAPE	12.021	11.829	12.059	11.927	11.851	14.718
MASE	1.631	1.585	1.623	1.613	1.599	2.408
OWA	0.87	0.851	0.869	0.861	0.855	1.172
Методы	**LightTS**	**DLinear**	**FEDformer**	**Stationary**	**Autoformer**	**Informer**
SMAPE	13.525	13.639	12.84	12.78	12.909	14.086
MASE	2.111	2.095	1.701	1.756	1.771	2.718
OWA	1.051	1.051	0.918	0.930	0.939	1.230

[5] Разработано автором.
[6] Разработано автором.

Конкурентоспособность предложенной модели была подтверждена сравнением с передовыми моделями, такими как PatchTST и TimesNet, где предложенная модель демонстрировала сопоставимую производительность. Кроме того, модель превзошла модели на основе многослойных перцептронов (MLP) в надежности производительности и оказалась более эффективной, чем модели на основе трансформеров, подтверждая потенциал RNN в анализе временных рядов.

Как показано на рисунке 2, предложенная модель, основанная на временных рядах RNN, обеспечивает надежную производительность (по сравнению с моделями, основанными на MLP) и эффективность (по сравнению с моделями, основанными на трансформерах).

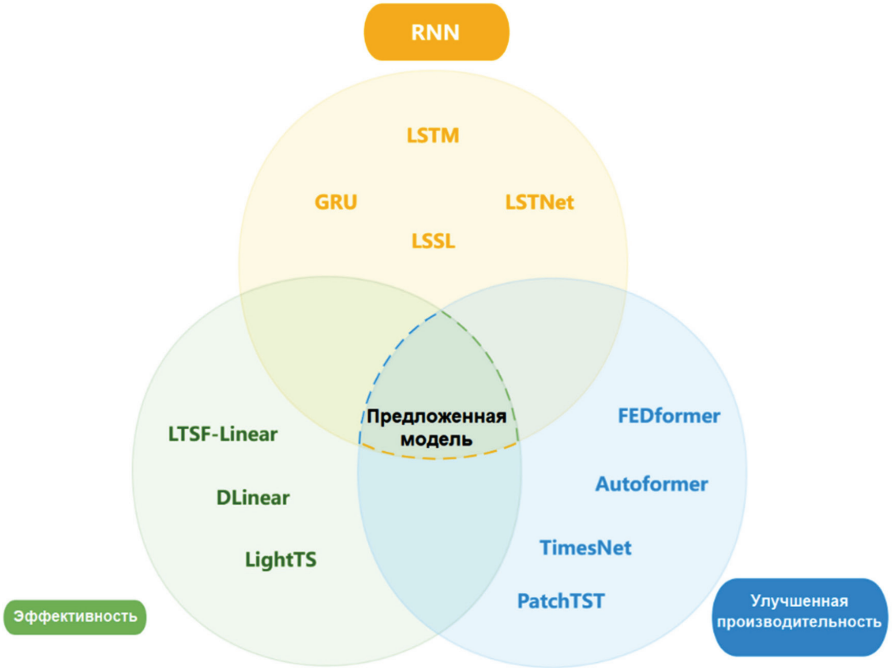

Рисунок 2 – Предложенная модель временных рядов на основе RNN[7]

Полученные результаты исследования включают:

1. Модель рекуррентных нейронных сетей (RNN) с линейной временной сложностью и оптимизированным использованием памяти.

2. Повышение производительности новой модели, сравнимой с передовыми методами, при существенном снижении вычислительной задержки и объема памяти.

Значительное улучшение результатов по сравнению с существующими моделями RNN, что подчеркивает потенциал применения RNN в анализе временных рядов.

Заключение

Предложенная модель рекуррентных нейронных сетей (RNN) с линейной временной сложностью и оптимизированным использованием памяти достигла производительности, сопоставимой с передовыми достижениями по нескольким временным рядам, с разной степенью успеха по сравнению с моделями, такими как PatchTST и TimesNet, на разных задачах. Самое заметное преимущество модели – ее линейная временная сложность и использование памяти, что позволяет быть эффективной, быстрой и иметь небольшой объем памяти. Это делает модель чрезвычайно конкурентоспособной для развертывания на конечных устройствах с ограниченными вычислительными ресурсами и объемами памяти. Кроме того, успех модели решает начальный вопрос о том, подходят ли рекуррентные нейронные сети для задач временных рядов. Эмпирический успех сети подчеркивает устойчивость этой архитектуры

[7] Разработано автором.

в анализе временных рядов. Возможность модели достигать конкурентоспособной производительности при решении вычислительных сложностей стимулирует дальнейшее исследование и инновации в подходах на основе RNN в области временных рядов. Будущие исследования должны позволить погрузиться глубже в улучшение эффективности и производительности архитектур RNN, нацеленных на разнообразные характеристики временных данных и их приложений.

Список литературы

1. *Kaplan J., McCandlish S., Henighan T., Brown T.B., Chess B., Child B., Gray S., Radford A., Wu J., Amodei D.* Scaling Laws for Neural Language Models. CoRR abs/2001.08361, 2020.
2. *Zhou H., Zhang S., Peng J., Zhang S., Li J., Xiong H., Zhang W.* Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. – 2021.
3. *Zhou T., Niu P., Wang X., Su, L., Jin R.* One fits all: power general time series analysis by pretrained, NeurIPS. – 2023.
4. *Zhou T., Ma Z., Wen Q., Wang X., Sun L., Jin R.* FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. – 2022.
5. *Tan Y., Xie L., Cheng X.* Neural Differential Recurrent Neural Network with Adaptive Time Steps. CoRR abs/2306.01674, 2023.
6. *Bergsma S., Zeyl T., Anaraki Z., Guo L.* C2FAR: Coarse-to-Fine Autoregressive Networks for Precise Probabilistic Forecasting // Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems NeurIPS, 2022 / Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, A. Oh, editors. – New Orleans, LA, USA, 2022.
7. *Peng B., Alcaide E., Anthony Q.G., Albalak A., Arcadinho S., Biderman S., Cao H., Cheng X., Chung M., Grella M., Kranthikiran G., He X., Hou H., Kazienko P., Kocoń J., Kong J., Koptyra B., Lau H., Mantri K.S., Mom F., Saito A., Tang X., Wang B., Wind J.S., Wozniak S., Zhang R., Zhang Z., Zhao Q., Zhou P., Zhu J., & Zhu R.* RWKV: Reinventing RNNs for the Transformer Era // Conference on Empirical Methods in Natural Language Processing, 2023. – DOI org/10.48550/arXiv.2305.13048.
8. *Sun Y., Dong L., Huang S., Ma S., Xia Y., Xue J., Wang J., Wei F.* Retentive network: A successor to transformer for large language models. ArXiv, abs/2307.08621, 2023.
9. *Li S., Jin X., Xuan Y., Zhou X., Chen W., Wang Y.-X., Yan X.* Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting // Advances in Neural Information Processing Systems (NeurIPS). – 2019. – Vol. 32.
10. *Oreshkin B., Carpo D., Chapados N., Bengio Y.* N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. – 2019.
11. *Zeng A., Chen M., Zhang L., Xu Q.* Are Transformers Effective for Time Series Forecasting? – 2022. – DOI 10.48550/arXiv.2205.13504.
12. *Vijay E., Jati A., Nguyen N., Sinthong G., Kalagnanam J.* Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting // ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – 2023.
13. *Das A., Kong W., Leach A., Mathur S., Sen R., Yu R.* Long-term forecasting with tide: Time-series dense encoder. – 2023.
14. *Zhang Y. & Wu R. & Dascalu S. & Harris F.* Multi-scale Transformer Pyramid Networks for Multivariate Time Series Forecasting. – 2023.
15. *Zhe L., Shiyi Q., Yiduo L., Zenglin X.* Revisiting Long-term Time Series Forecasting: An Investigation on Linear Mapping. – 2023.
16. *Yuxin W., Kaiming H.* Group Normalization // International Journal of Computer Vision. – 2020. – DOI 10.1007/s11263-019-01198-w.
17. *Wu H., Hu T., Liu Y., Zhou H., Wang J., Long M.* TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. – 2023.
18. *Nie Y., Nguyen N.H., Sinthong P., Kalagnanam J.* A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. – 2023.

References

1. *Kaplan J., McCandlish S., Henighan T., Brown T.B., Chess B., Child B., Gray S., Radford A., Wu J., Amodei D.* Scaling Laws for Neural Language Models. CoRR abs/2001.08361, 2020.
2. *Zhou H., Zhang S., Peng J., Zhang S., Li J., Xiong H., Zhang W.* Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. – 2021.
3. *Zhou T., Niu P., Wang X., Su, L., Jin R.* One fits all: power general time series analysis by pretrained, NeurIPS. – 2023.
4. *Zhou T., Ma Z., Wen Q., Wang X., Sun L., Jin R.* FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. – 2022.
5. *Tan Y., Xie L., Cheng X.* Neural Differential Recurrent Neural Network with Adaptive Time Steps. CoRR abs/2306.01674, 2023.
6. *Bergsma S., Zeyl T., Anaraki Z., Guo L.* C2FAR: Coarse-to-Fine Autoregressive Networks for Precise Probabilistic Forecasting // Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems NeurIPS, 2022 / Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, A. Oh, editors. – New Orleans, LA, USA, 2022.
7. *Peng B., Alcaide E., Anthony Q.G., Albalak A., Arcadinho S., Biderman S., Cao H., Cheng X., Chung M., Grella M., Kranthikiran G., He X., Hou H., Kazienko P., Kocoń J., Kong J., Koptyra B., Lau H., Mantri K.S., Mom F., Saito A., Tang X., Wang B., Wind J.S., Wozniak S., Zhang R., Zhang Z., Zhao Q., Zhou P., Zhu J., & Zhu R.* RWKV: Reinventing RNNs for the Transformer Era // Conference on Empirical Methods in Natural Language Processing, 2023. – DOI org/10.48550/arXiv.2305.13048.
8. *Sun Y., Dong L., Huang S., Ma S., Xia Y., Xue J., Wang J., Wei F.* Retentive network: A successor to transformer for large language models. ArXiv, abs/2307.08621, 2023.
9. *Li S., Jin X., Xuan Y., Zhou X., Chen W., Wang Y.-X., Yan X.* Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting // Advances in Neural Information Processing Systems (NeurIPS). – 2019. – Vol. 32.
10. *Oreshkin B., Carpo D., Chapados N., Bengio Y.* N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. – 2019.
11. *Zeng A., Chen M., Zhang L., Xu Q.* Are Transformers Effective for Time Series Forecasting? – 2022. – DOI 10.48550/arXiv.2205.13504.
12. *Vijay E., Jati A., Nguyen N., Sinthong G., Kalagnanam J.* Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting // ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – 2023.
13. *Das A., Kong W., Leach A., Mathur S., Sen R., Yu R.* Long-term forecasting with tide: Time-series dense encoder. – 2023.
14. *Zhang Y. & Wu R. & Dascalu S. & Harris F.* Multi-scale Transformer Pyramid Networks for Multivariate Time Series Forecasting. – 2023.
15. *Zhe L., Shiyi Q., Yiduo L., Zenglin X.* Revisiting Long-term Time Series Forecasting: An Investigation on Linear Mapping. – 2023.
16. *Yuxin W., Kaiming H.* Group Normalization // International Journal of Computer Vision. – 2020. – DOI 10.1007/s11263-019-01198-w.
17. *Wu H., Hu T., Liu Y., Zhou H., Wang J., Long M.* TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. – 2023.
18. *Nie Y., Nguyen N.H., Sinthong P., Kalagnanam J.* A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. – 2023.