Исследование влияния модульной гибридизации на точность и эффективность трансформеров для долгосрочного прогнозирования временных рядов *

Очкин Н.В.
email@email

Зубарев К.М.
email@email

Аннотация

Задача долгосрочного прогнозирования многомерных временных рядов остается сложной: в реальных данных наблюдается ряд проблемных эффектов – среди них, краткосрочные мотивы (скачки, ступени), крайне долгосрочные и межрядовые зависимости, выраженная нестационарность (сдвиги тренда/уровня, мультисезонность, гетероскедастичность). В данной работе мы предлагаем расширить слой эмбеддинга в модели Informer [21] компактным двухслойным сверточным блоком с целью повышения эффективности извлечения локальных паттернов, внедрить модуль декомпозиции ряда из Autoformer [19] для явного разделения трендовых и сезонных компонент и заменить механизм ProbSparse на линейное внимание FAVOR+ (Performer [5]) для учёта глобальных зависимостей при низких вычислительных затратах. Эксперименты на стандартных бенчмарках демонстрируют снижение ошибок на длинных горизонтах прогнозирования в ряде настроек. Абляционные исследования указывают на вклад каждого из модулей, хотя эффект не универсален для всех горизонтов. На рассмотренных конфигурациях наблюдается благоприятный компромисс между точностью и затратами времени/памяти.

Keywords LSTF · Long Sequence Time-series Forecasting · Transformer · Informer · Performer · Autoformer

1 Введение

Долгосрочное прогнозирование временных рядов является одной из ключевых задач во многих областях – от энергетики и финансов до транспорта и здравоохранения. Однако реальные многомерные ряды редко обладают простой структурой: среди прочего, они сочетают краткосрочные локальные паттерны, очень длинные и межрядовые зависимости, а также выраженную нестационарность, включая сдвиги тренда, мультисезонность и гетероскедастичность. Эти свойства затрудняют обучение моделей и ухудшают их способность к экстраполяции на длительных горизонтах; классические статистические методы (ARIMA, VAR, экспоненциальное сглаживание и др.) обычно эффективны лишь на умеренных горизонтах и почти стационарных данных.

С момента публикации архитектура Transformer [18] завоевала широкое признание. Однако у нее есть несколько серьезных проблем, которые усложняют работу с длинными временными последовательностями (LSTF). Последующие исследования предложили различные методы решения данных и связанных с ними проблем: Informer [21] вводит разреженное ProbSparse внимание и схему distilling для снижения квадратичных затрат по длине последовательности; Autoformer [19] дополняет архитектуру явной декомпозицией ряда на тренд и сезонный остаток и модулем AutoCorrelation для устойчивого прогнозирования на длинных горизонтах; Performer [5] предлагает FAVOR+ – несмещённую аппроксимацию softmax-ядра с линейной по длине последовательности сложностью. Тем не менее существующие архитектуры сталкиваются с рядом ограничений:

*Исходный код, конфигурации и скрипты обучения доступны: https://github.com/namenick91/Convformer

необходимость одновременного учёта локальных и глобальных закономерностей, высокая чувствительность к нестационарности и ограниченная масштабируемость по длине входной последовательности.

В данной работе мы предлагаем архитектуру Convformer, которая в единой схеме объединяет три взаимодополняющих индуктивных смещения, специально ориентированных на решение перечисленных проблем: сверточный входной блок ConvStem для выделения локальных паттернов, линейное внимание FAVOR+ для масштабируемого моделирования глобальных зависимостей и Autoformer-подобную декомпозицию ряда после каждого блока self-attention.

Основной вклад данной работы можно резюмировать следующим образом:

- Вводим расширенный сверточный входной блок для кодирования краткосрочных мотивов (скачки, ступени, импульсные всплески) и стабилизации статистик нестационарных входов.
- Для моделирования глобальных связей применяем механизм внимания FAVOR+ с линейными затратами по времени и памяти, что обеспечивает масштабируемое моделирование дальних зависимостей.
- Встраиваем Autoformer-подобную [19] декомпозицию после каждого блока self-attention, которая пропускает низкочастотный тренд по остаточным соединениям и передаёт более стационаризированный поток данных в механизм внимания.
- Эксперименты на стандартных бенчмарках демонстрируют снижение ошибок на длительных горизонтах прогнозирования; абляционные исследования подтверждают вклад каждого модуля, а сравнительный анализ показывает благоприятный баланс между точностью и вычислительными затратами.

2 Обозначения и предварительные сведения

Приведём формальное определение задачи долгосрочного прогнозирования временных рядов (LSTF). Рассматривается дискретный многомерный временной ряд $\{\mathbf{x}_t\}_{t=1}^{T},\ \mathbf{x}_t \in \mathbb{R}^{C_{\text{in}}}$. Пусть $\mathbf{y}_t \in \mathbb{R}^{C_{\text{out}}}$ обозначает вектор таргетных компонент $\mathbf{x}_t$, причём $C_{\text{out}} \le C_{\text{in}}$. Для каждого момента времени t, удовлетворяющего $L_x \le t \le T - L_y$, определим входное окно наблюдений $\mathcal{X}_t = \{\mathbf{x}_{t-L_x+1}, \dots, \mathbf{x}_t\}$, и соответствующую ему последовательность будущих таргетных значений $\mathcal{Y}_t = \{\mathbf{y}_{t+1}, \dots, \mathbf{y}_{t+L_y}\}$. Здесь и в дальнейшем: t – глобальный индекс времени, L_x, L_y – длины входной и выходной последовательностей, $C_{\text{in}}, C_{\text{out}}$ – число входных и прогнозируемых признаков (измеряемых величин) в каждый момент времени соответственно. Дальнейшие рассуждения одинаково применимы к многомерному и одномерному случаям; различие заключается лишь в выборе C_{out} и в том, какие компоненты $\mathbf{x}_t$ считаются таргетами.

Задача LSTF формулируется как задача обучения параметризованного отображения $f_\theta : \mathbb{R}^{L_x \times C_{\text{in}}} \to \mathbb{R}^{L_y \times C_{\text{out}}}$, которое по входному окну наблюдений $\mathcal{X}_t$ восстанавливает соответствующую ему последовательность будущих значений $\mathcal{Y}_t$:

$$\hat{\mathcal{Y}}_t = f_\theta(\mathcal{X}_t) \approx \mathcal{Y}_t.$$

Задача LSTF предполагает предсказание далёкого будущего, то есть больших значений L_y; при этом размерность признаков не ограничивается одномерным случаем. Пусть обучающая выборка состоит из набора окон $\mathcal{D} = \{(\mathcal{X}_t, \mathcal{Y}_t)\}_{t \in \mathcal{T}}$, где $\mathcal{T}$ – множество индексов времени, для которых формируются пары $(\mathcal{X}_t, \mathcal{Y}_t)$. Обучение модели f_θ проводится в постановке контролируемого обучения путём минимизации среднеквадратичной функции потерь:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{1}{L_y \cdot C_{\text{out}}} \left\| f_\theta(\mathcal{X}_t) - \mathcal{Y}_t \right\|_F^2,$$

где $\|\cdot\|_F$ обозначает норму Фробениуса. В дальнейшем под f_θ рассматриваются как базовые трансформерные архитектуры Informer, Performer и Autoformer, так и предлагаемые в данной работе модификации. Все модели обучаются в единой постановке, отличаясь лишь внутренней реализацией отображения f_θ при фиксированных $L_x, L_y, C_{\text{in}}, C_{\text{out}}$.

2.1 Классический механизм внимания

Пусть $\mathbf{Q} \in \mathbb{R}^{L_Q \times d_k}$, $\mathbf{K} \in \mathbb{R}^{L_K \times d_k}$, $\mathbf{V} \in \mathbb{R}^{L_K \times d_v}$ – промежуточные представления входных данных, строки которых можно интерпретировать как запросы, ключи и значения непрерывной словарной структуры данных

соответственно [18], где L_Q и L_K обозначают длину последовательностей запросов и ключей/значений соответственно, d_k – размерность запросов и ключей, а d_v – размерность значений. Двунаправленное (bidirectional, или неориентированное, non-directional [6]) внимание на основе скалярного произведения имеет вид:

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \in \mathbb{R}^{L_Q \times d_v}, \tag{1}$$

где $\mathrm{softmax}(\cdot)$ применяется построчно.

Временная и пространственная сложность вычисления (1) равны $O(L_Q L_K(d_k + d_v))$ и $O(L_Q L_K + L_Q d_k + L_K d_k + L_K d_v)$ соответственно (при $L_Q = L_K = L$ и $d_v = d_k$ получаем $O(L^2 d_k)$ по времени и $O(L^2 + L d_k)$ по памяти). Поэтому механизм внимания на основе скалярного произведения (1) в принципе несовместим с end-to-end-обработкой длинных последовательностей. Двунаправленное внимание используется в self-attention энкодера и в cross-attention энкодер-декодера в архитектурах Seq2Seq.

Другой важный тип внимания – однонаправленное (unidirectional) внимание:

$$\mathrm{Att}_{\rightarrow}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V} \in \mathbb{R}^{L_Q \times d_v},$$

где $\mathbf{M} \in \mathbb{R}^{L_Q \times L_K}$ – маска, элементы которой задают, какие логиты участвуют в нормализации (элементы, равные $-\infty$, полностью подавляют соответствующие ключи для данного запроса).

Однонаправленное внимание используется в self-attention декодера в архитектурах Seq2Seq. В частности, в случае self-attention декодера, когда $L_Q = L_K = L$, обычно используют нижнетреугольную каузальную маску, задаваемую правилом $M_{ij} = 0$ при $j \le i$ и $M_{ij} = -\infty$ при $j > i$.

На практике в трансформерных архитектурах используется стандартный multi-head attention с h головами [18]. В настоящей работе мы также используем multi-head attention; для простоты дальнейшего изложения ниже приводим запись для одной головы, а multi-head вариант получается параллельным применением механизма по головам с последующей конкатенацией результатов по признаковому измерению.

Механизм ProbSparse self-attention В Informer [21] предлагается механизм ProbSparse self-attention, направленный на снижение квадратичной сложности стандартного внимания до логарифмического. Основная идея заключается в том, что для большинства запросов вклад большинства ключей в распределение attention оказывается пренебрежимо малым. В ProbSparse вводится статистика «важности» запроса, оценивающая разброс логитов по ключам, и для вычисления attention явно используются только запросы с наибольшей важностью. Это позволяет уменьшить вычислительные затраты по времени и памяти при сохранении основных пиков распределения внимания, однако вносит смещение за счёт явной выборки ключей и чувствительности к структуре данных.

2.2 Ядерное внимание и random features

Выражение (1) можно эквивалентно переписать в ядерной форме:

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) = \sum_{j=1}^{L_K} \frac{k(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{l=1}^{L_K} k(\mathbf{q}_i, \mathbf{k}_l)}\mathbf{v}_j, \tag{2}$$

где $\mathbf{q}_i \in \mathbb{R}^{d_k}$ – i-я строка матрицы $\mathbf{Q}$, $\mathbf{k}_j \in \mathbb{R}^{d_k}$ и $\mathbf{v}_j \in \mathbb{R}^{d_v}$ – j-е строки матриц $\mathbf{K}$ и $\mathbf{V}$ соответственно, а $k(\mathbf{q}_i, \mathbf{k}_j) \coloneqq \exp(\mathbf{q}_i \mathbf{k}_j^\top / \sqrt{d_k})$ задаёт экспоненциальное (softmax) ядро.

Следуя Performer [5], без ограничения общности опустим $\sqrt{d_k}$-нормировку, поскольку можем эквивалентно перенормировать входные ключи и запросы, и далее будем работать с softmax-ядром: $\mathrm{SM}(\mathbf{x}, \mathbf{y}) \coloneqq \exp(\mathbf{x}^\top \mathbf{y})$. Следуя подходу случайных признаков (random features) [16], предположим, что для ядра k существует случайное скалярное отображение вида:

$$k(\mathbf{x}, \mathbf{y}) = \mathbb{E}_\omega[\varphi_\omega(\mathbf{x})\varphi_\omega(\mathbf{y})], \qquad \varphi_\omega : \mathbb{R}^{d_k} \to \mathbb{R},$$

так что при фиксированном наборе случайных параметров $\omega_1, \ldots, \omega_m \in \mathbb{R}^{d_k}$ можно построить эмпирическую карту признаков $\phi(\mathbf{x}) = \frac{1}{\sqrt{m}}(\varphi_{\omega_1}(\mathbf{x}), \ldots, \varphi_{\omega_m}(\mathbf{x})) \in \mathbb{R}^m$, для которой: $k(\mathbf{x}, \mathbf{y}) \approx \phi(\mathbf{x})^\top \phi(\mathbf{y})$. Тогда мы можем аппроксимировать ядро в (2) как: $k(\mathbf{q}_i, \mathbf{k}_j) \approx \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j)$. Подставляя эту аппроксимацию в (2), получаем линейную по длинам последовательностей L_Q, L_K схему вычисления внимания:

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) \approx \frac{\phi(\mathbf{q}_i)^\top \left(\sum_{j=1}^{L_K} \phi(\mathbf{k}_j) \mathbf{v}_j^\top \right)}{\phi(\mathbf{q}_i)^\top \left(\sum_{l=1}^{L_K} \phi(\mathbf{k}_l) \right)}. \tag{3}$$

Это позволяет обходиться без явного построения матрицы $\mathcal{K} \in \mathbb{R}^{L_Q \times L_K}$, $\mathcal{K}_{ij} = k(\mathbf{q}_i, \mathbf{k}_j)$. При фиксированной размерности карты признаков r вычислительная сложность становится линейной по L: $O((L_Q + L_K) r (d_k + d_v))$ вместо квадратичной $O(L_Q L_K (d_k + d_v))$ у классического внимания.

Как показано в [5, 16], широкий класс ядер можно реализовать через карты вида:

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}} \big(f_1(\omega_1^\top \mathbf{x}), \ldots, f_1(\omega_m^\top \mathbf{x}), \ldots, f_l(\omega_1^\top \mathbf{x}), \ldots, f_l(\omega_m^\top \mathbf{x}) \big) \in \mathbb{R}^r,$$

где $f_1, \ldots, f_l : \mathbb{R} \to \mathbb{R}$, $h : \mathbb{R}^{d_k} \to \mathbb{R}$, $\omega_1, \ldots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$, для некоторого распределения $\mathcal{D} \in \mathcal{P}(\mathbb{R}^{d_k})$, и $r = lm$.

В частности, для несмещённой аппроксимации ядра $\mathrm{SM}(\mathbf{x}, \mathbf{y})$ можно использовать тригонометрическую карту случайных признаков [4, 5]: $h(\mathbf{x}) = \exp(\frac{||\mathbf{x}||^2}{2})$, $l = 2$, $f_1 = \sin$, $f_2 = \cos$, $\omega_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_k})$. Назовем ее $\hat{\mathrm{SM}}_m^{\mathrm{trig}}$.

FAVOR+ как схема аппроксимации softmax-внимания В работе [5] показано, что аппроксимация вида $\hat{\mathrm{SM}}_m^{\mathrm{trig}}$ из-за наличия потенциально отрицательных компонентов приводит к нестабильному поведению модели. Авторы предлагают более устойчивый механизм аппроксимации softmax-ядра через карту положительных случайных признаков (positive random feature map), снижающий дисперсию аппроксимации [5]:

$$\phi(\mathbf{x}) = \frac{\exp(-\frac{||\mathbf{x}||^2}{2})}{\sqrt{2m}} (e^{\omega_1^\top \mathbf{x}}, \ldots, e^{\omega_m^\top \mathbf{x}}, e^{-\omega_1^\top \mathbf{x}}, \ldots, e^{-\omega_m^\top \mathbf{x}}), \quad \omega_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_k}). \tag{4}$$

Дополнительно используется ортогонализация случайных признаков (orthogonal random features). При изотропном распределении $\mathcal{D}$ (на практике чаще всего берут $\mathcal{N}(\mathbf{0}, \mathbf{I}_{d_k})$) ортогональные векторы ω_i можно получать с помощью стандартной процедуры ортогонализации Грама-Шмидта, сохраняя несмещённость оценок и ещё сильнее снижая дисперсию аппроксимации softmax-ядра.

Подставляя полученное отображение в общую схему линейного внимания (3), получаем механизм FAVOR+ (Fast Attention Via positive Orthogonal Random features) для softmax-внимания, обладающий линейной по длине последовательности сложностью и теоретическими гарантиями на точность аппроксимации матрицы внимания.

2.3 Представление входных данных

По аналогии с Informer [21] используем единое представление входного окна, объединяющее численные значения ряда, локальный позиционный контекст и глобальные календарные метки. Пусть $\mathcal{X}_t \in \mathbb{R}^{L_x \times C_{\text{in}}}$ – окно наблюдений, а $\mathcal{Z}_t \in \mathbb{Z}^{L_x \times p}$ – матрица дискретных временных меток для каждого момента времени (месяц, день, день недели, час, при необходимости – минута). Размерность скрытого пространства модели обозначим через d_{model}. Тогда эмбеддинговое представление $\mathcal{H}_{t,0} \in \mathbb{R}^{L_x \times d_{\text{model}}}$ (блок ConvEmbedding, см. рис. 3.2) строится как сумма трёх слагаемых:

$$(\mathcal{X}_t, \mathcal{Z}_t) \;\mapsto\; \mathcal{H}_{t,0} = \mathrm{CS}(\mathcal{X}_t) + \mathrm{PE} + \mathrm{TE}(\mathcal{Z}_t),$$

где $\mathrm{CS} = \mathrm{ConvStem} : \mathbb{R}^{L_x \times C_{\text{in}}} \to \mathbb{R}^{L_x \times d_{\text{model}}}$ – сверточная проекция значений (value embedding) (см. раздел 3); $\mathrm{PE} \in \mathbb{R}^{L_x \times d_{\text{model}}}$ – матрица позиционного кодирования; $\mathrm{TE} : \mathbb{Z}^{L_x \times p} \to \mathbb{R}^{L_x \times d_{\text{model}}}$ – темпоральное кодирование.

Позиционное кодирование Используем синусоидальное позиционное кодирование, совпадающее с классическим Transformer [18]. Для позиции $\text{pos} \in \{1, \dots, L_x\}$ и индекса $i \in \{0, \dots, \lfloor d_{\text{model}}/2 \rfloor - 1\}$ определим:

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{\,2i/d_{\text{model}}}}\right), \qquad \text{PE}(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{\,2i/d_{\text{model}}}}\right).$$

Матрица $\text{PE} \in \mathbb{R}^{L_x \times d_{\text{model}}}$ фиксируется заранее и не обучается.

Темпоральное кодирование Для каждого временного шага $i = 1, \dots, L_x$ храним набор из p дискретных календарных признаков: месяц, день, день недели, час и, при необходимости, минуту. Сведём их в матрицу $\mathcal{Z}_t \in \mathbb{Z}^{L_x \times p}$, где $\mathcal{Z}_t[i,k] \in \{0, \dots, S_k - 1\}$ – индекс категории k-го календарного признака в момент времени, соответствующий позиции i, а S_k – размер словаря для k-го признака. Каждому типу календарной метки k ставится в соответствие обучаемая матрица эмбеддингов $E^{(k)} \in \mathbb{R}^{S_k \times d_{\text{model}}}$, которую удобно интерпретировать как отображение $E^{(k)} : \{0, \dots, S_k - 1\} \to \mathbb{R}^{d_{\text{model}}}$: по индексу $\mathcal{Z}_t[i,k]$ возвращается вектор размерности d_{model}.

Отображение темпорального кодирования $\text{TE} : \mathbb{Z}^{L_x \times p} \to \mathbb{R}^{L_x \times d_{\text{model}}}$ определим покомпонентно:

$$\text{TE}(\mathcal{Z}_t)[i,:] = \sum_{k=1}^{p} E^{(k)}\big[\mathcal{Z}_t[i,k], :\big], \qquad i = 1, \dots, L_x.$$

Таким образом, темпоральный вектор для каждого шага получается суммой эмбеддингов всех доступных календарных признаков (месяц, день, день недели, час, минута).

3 Методология

Повышение эффективности извлечения локальных паттернов Для усиления способности модели к распознаванию краткосрочных закономерностей мы предлагаем заменить стандартный механизм представления значений в блоке с обработкой входных данных в Informer [21] на сверточный блок ConvStem, тогда как позиционное и темпоральное кодирование остаются неизменными (см. раздел 2.3). Данный блок сочетает проекцию входных признаков через точечную свёртку с последующими операциями широкой и глубинной свёрток, дополненных нормализацией и нелинейностью. Такое построение позволяет модели фиксировать повторяющиеся локальные мотивы и вариации формы сигналов непосредственно на этапе встраивания, ещё до применения механизмов внимания. В результате глобальное внимание может быть сфокусировано преимущественно на долговременных зависимостях, тогда как локальная динамика эффективно извлекается специализированным сверточным модулем.

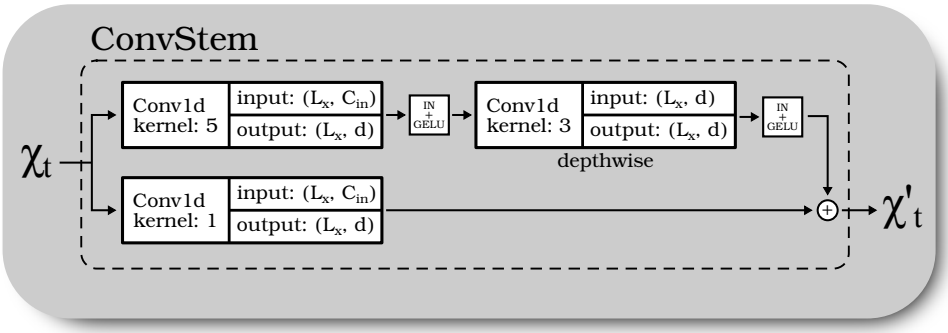


Рис. 3.1: Схема блока ConvStem. Здесь: $\mathcal{X}_t \in \mathbb{R}^{L_x \times C_{\text{in}}}$, $\mathcal{X}'_t \in \mathbb{R}^{L_x \times d}$, где $d = d_{\text{model}}$ – размерность модели; IN – нормализация по экземплярам (Instance Normalization [17]). Псевдокод приведен в алгоритме 1.

Масштабируемое моделирование глобальных зависимостей В качестве механизма внимания предлагается использовать FAVOR+ (Performer [5]) вместо ProbSparse (оригинально применяемого в Informer [21]). Подобно тому, как в оригинальном Informer [21] классический механизм полного внимания FullAttention [18] был заменен на ProbSparse исключительно для вычисления self-attention (механизм самовнимания) в слоях коди-

ровщика и декодера, в нашей модификации ProbSparse заменяется на FAVOR+ в тех же местах, тогда как cross-attention останется реализованным через полное внимание (FullAttention). Данную замену мы мотивируем тем, что FAVOR+ обеспечивает несмещённую аппроксимацию softmax-ядра с линейными по длине последовательности затратами по времени и памяти [5], что гарантирует предсказуемую и масштабируемую работу на очень длинных последовательностях в условиях ограниченных ресурсов GPU. При этом точность аппроксимации управляется размерностью карты случайных признаков r: при меньших значениях достигается высокая скорость, при больших – более точное восстановление распределения внимания. Такая настраиваемость делает механизм универсальным инструментом, позволяющим варьировать баланс между эффективностью и качеством. Кроме того, отказ от ручной выборки top-u запросов позволяет модели учитывать любые глобальные зависимости, а не только наиболее сильные периодические, что важно для корректного описания непериодических скачков в реальных временных рядах.

Для практической реализации FAVOR+ мы используем модуль `Favor` из библиотеки `fast-transformers` [11], который представляет собой реализацию положительных ортогональных случайных признаков для softmax-ядра (формула (4)). Во всех экспериментах размерность карты случайных признаков фиксирована и равна $r = 256$.

Явное разделение тренда и сезонности Следуя Autoformer [19], мы интегрируем в архитектуру механизм декомпозиции временных рядов (SeriesDecomp) – внутреннюю операцию, которая постепенно извлекает долгосрочную трендовую (трендово-циклическую) составляющую из промежуточных скрытых представлений. Для скрытого представления $\mathcal{X}_t \in \mathbb{R}^{L_x \times d_{\text{model}}}$ декомпозиция определяется как:

$$\mathcal{X}_t^{(\text{tr})} = \text{MA}_k(\mathcal{X}_t),$$
$$\mathcal{X}_t^{(\text{se})} = \mathcal{X}_t - \mathcal{X}_t^{(\text{tr})},$$

где $\mathcal{X}_t^{(\text{tr})}, \mathcal{X}_t^{(\text{se})} \in \mathbb{R}^{L_x \times d_{\text{model}}}$ – трендово-циклическая и сезонная составляющие соответственно; $\text{MA}_k(\cdot)$ – оператор скользящего среднего (simple moving average) с окном длины k, единичным шагом и постоянным дополнением на границах. В дальнейшем будем использовать обозначение

$$(\mathcal{X}_t^{(\text{tr})}, \mathcal{X}_t^{(\text{se})}) = \text{SeriesDecomp}(\mathcal{X}_t)$$

для краткой записи блока декомпозиции.

В архитектуре нашей модели блок SeriesDecomp многократно применяется к скрытым представлениям, обеспечивая их разделение на два канала: низкочастотный тренд и остаточную компоненту, содержащую более стационарные сезонные и случайные колебания. Трендовая часть передаётся по остаточному пути, обеспечивая стабильность прогнозов при сдвигах уровня, тогда как внимание применяется к сезонной компоненте, где наиболее выражены периодические и локальные закономерности. Такая стратегия снижает вариативность обучения, повышает способность к экстраполяции за пределы обучающего диапазона и улучшает интерпретируемость результатов за счёт раздельного моделирования трендовой и сезонной составляющих.

3.1 Энкодер

Подобно тому, как это реализовано в Autoformer [19], энкодер в нашей модели сосредоточивается на моделировании сезонной составляющей (см. рис. 3.2). Его выход содержит прошлую сезонную информацию и используется в качестве входа для перекрёстного внимания в декодере, что позволяет последнему уточнять результаты прогнозирования.

Пусть N – количество слоёв в энкодере, $\mathcal{H}_{t,0,\text{enc}}$ – эмбеддинговое представление входного окна $\mathcal{X}_t$. Чтобы не перегружать обозначения, зафиксируем момент времени и опустим индекс t, т.е. $\mathcal{X}_t, \mathcal{H}_{t,0,\text{enc}} \Leftrightarrow \mathcal{X}, \mathcal{H}_{0,\text{enc}}$. Тогда работу энкодера на слое n можно описать следующим образом:

$$\begin{aligned}
\big(\mathcal{H}_{n,\text{enc}}^{(\text{self,se})}, \mathcal{H}_{n,\text{enc}}^{(\text{self,tr})}\big) &= \text{SeriesDecomp}\big(\text{FAVOR}_{\leftrightarrow}(\mathcal{Z}_{n-1}^{(\text{enc})}) + \mathcal{Z}_{n-1}^{(\text{enc})}\big), && n \in \{1, \dots, N\},\\
\big(\mathcal{H}_{n,\text{enc}}^{(\text{ff,se})}, \mathcal{H}_{n,\text{enc}}^{(\text{ff,tr})}\big) &= \text{SeriesDecomp}\big(\text{FF}(\mathcal{H}_{n,\text{enc}}^{(\text{self,se})}) + \mathcal{H}_{n,\text{enc}}^{(\text{self,se})}\big), && n \in \{1, \dots, N\},\\
\mathcal{H}_{n,\text{enc}}^{(\text{dist})} &= \text{Distilling}\big(\mathcal{H}_{n,\text{enc}}^{(\text{ff,se})}\big), && n \in \{1, \dots, N-1\}.
\end{aligned}$$

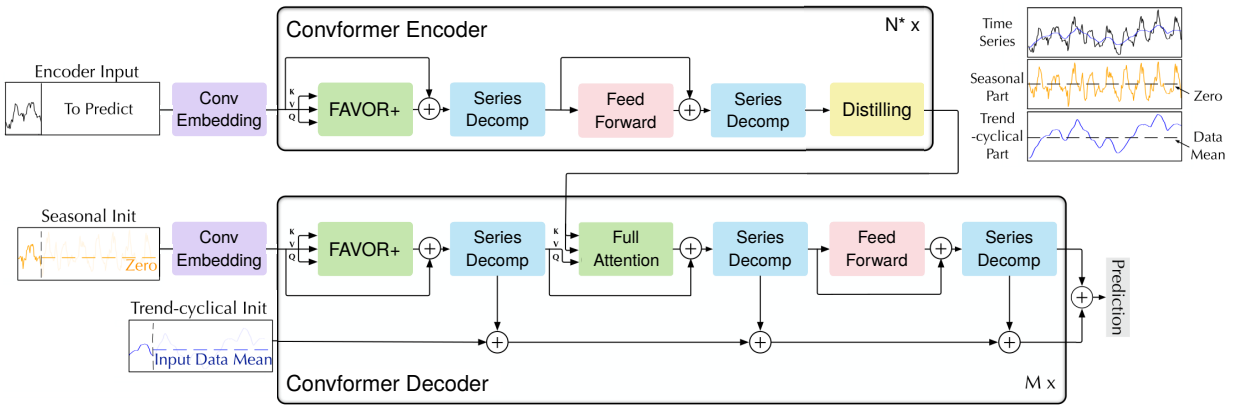


Рис. 3.2: Архитектура Convformer. Входные данные (для декодера – их сезонная составляющая) проходят через блок ConvEmbedding (фиолетовый блок), состоящий из ConvStem, позиционного и темпорального кодирований (подробнее см. в разделе 2.3). Энкодер устраняет долгосрочную трендово-циклическую составляющую с помощью блоков декомпозиции рядов (синие блоки) и сосредотачивается на моделировании сезонных компонент. Блок дистилляции N^* в энкодере (жёлтый блок) включён во все слои, кроме последнего. Декодер постепенно накапливает трендовую составляющую, извлекаемую из скрытых переменных. Прошлая сезонная информация, полученная от энкодера, используется в классическом (полном) механизме внимания (см. раздел 2.1) (центральный зелёный блок в декодере).

где $(\mathcal{Z}_\nu^{(\mathrm{enc})})_{\nu=0}^{N-1} \coloneqq \left(\mathcal{H}_0^{(\mathrm{enc})}, \mathcal{H}_{1,\mathrm{enc}}^{(\mathrm{dist})}, \ldots, \mathcal{H}_{N-1,\mathrm{enc}}^{(\mathrm{dist})}\right)$, на слое N дистилляция не применяется.

Здесь $\mathcal{H}_{n,\mathrm{enc}}^{(i,j)}$, $j \in \{\mathrm{se}, \mathrm{tr}\}$, $i \in \{\mathrm{self}, \mathrm{ff}\}$ – сезонные и трендовые компоненты после соответствующих внутренних подпроцессов (двунаправленного self-attention, feed-forward) соответственно; $\mathcal{H}_{n,\mathrm{enc}}^{(\mathrm{dist})}$ – результат блока дистилляции (см. раздел 3.1) в слое n; $\mathrm{FF}(\cdot)$ обозначает позиционно-независимую двухслойную полносвязную feed-forward сеть; $\mathrm{FAVOR}(\cdot)$ – механизм внимания FAVOR+ [5] (см. раздел 3). Трендовые компоненты в слоях энкодера явно не используются.

Дистилляция внимания Следуя Informer [21], используем дистилляцию внимания для выделения представлений с доминирующими признаками и формирования более сфокусированной карты признаков в следующем слое. Для произвольного скрытого представления $\mathcal{H}_n$ на слое n:

$$\mathcal{H}_n^{(\mathrm{dist})} = \mathrm{MaxPool}(\mathrm{ELU}(\mathrm{BN}(\mathrm{Conv1d}(\mathcal{H}_n)))),$$

где $\mathrm{Conv1d}$ обозначает одномерную свертку по времени (с ядром размера 3), BN – нормализацию по батчам ($\mathrm{BatchNorm1d}$ [10]), ELU – функцию активации, а $\mathrm{MaxPool}$ – одномерное максимальное объединение по времени, уменьшающее длину последовательности. Для краткой записи блока используем обозначение $\mathcal{H}_n^{(\mathrm{dist})} = \mathrm{Distilling}(\mathcal{H}_n)$.

3.2 Декодер

Подобно Autoformer [19], на вход декодеру одновременно подаются как сезонная компонента $\mathcal{X}_{t,\mathrm{dec}}^{(\mathrm{se})} \in \mathbb{R}^{(\frac{L_x}{2}+L_y)\times C_{\mathrm{in}}}$, так и трендово-циклическая $\mathcal{X}_{t,\mathrm{dec}}^{(\mathrm{tr})} \in \mathbb{R}^{(\frac{L_x}{2}+L_y)\times C_{\mathrm{in}}}$, формируемые из входного окна $\mathcal{X}_t$. Аналогично разделу с энкодером (см. раздел 3.1), опустим индекс t. Формально процесс инициализации декодера можно описать следующим образом:

$$\begin{aligned}
(\mathcal{X}^{(\mathrm{se})}, \mathcal{X}^{(\mathrm{tr})}) &= \mathrm{SeriesDecomp}(\mathcal{X}), \\
\mathcal{X}_{\mathrm{dec}}^{(\mathrm{se})} &= \mathrm{Concat}(\mathcal{X}_{\{\frac{L_x}{2}:L_x\}}^{(\mathrm{se})}, \mathcal{X}_0), \\
\mathcal{X}_{\mathrm{dec}}^{(\mathrm{tr})} &= \mathrm{Concat}(\mathcal{X}_{\{\frac{L_x}{2}:L_x\}}^{(\mathrm{tr})}, \bar{\mathcal{X}}),
\end{aligned}$$

где $\mathcal{X}^{(\mathrm{se})}_{\{\frac{L_x}{2}:L_x\}}, \mathcal{X}^{(\mathrm{tr})}_{\{\frac{L_x}{2}:L_x\}} \in \mathbb{R}^{\frac{L_x}{2}\times C_{\mathrm{in}}}$ обозначают последние $\frac{L_x}{2}$ элементов сезонной и трендовой составляющих входной последовательности $\mathcal{X}$ соответственно, а $\mathcal{X}_0, \bar{\mathcal{X}} \in \mathbb{R}^{L_y\times C_{\mathrm{in}}}$ – «заглушки», заполненные, соответственно, нулями и средним по времени исходной последовательности $\mathcal{X}$.

Пусть M – количество слоёв в декодере, $\mathcal{H}_{0,\mathrm{dec}}$ – эмбеддинговое представление входной сезонной последовательности $\mathcal{X}^{(\mathrm{se})}_{\mathrm{dec}}$. Тогда работу декодера на слое m, $m \in \{1,\dots,M\}$ можно описать следующим образом:

$$\begin{aligned}
\big(\mathcal{H}^{(\mathrm{self,se})}_{m,\mathrm{dec}}, \mathcal{H}^{(\mathrm{self,tr})}_{m,\mathrm{dec}}\big) &= \mathrm{SeriesDecomp}\big(\mathrm{FAVOR}_{\rightarrow}(\mathcal{Z}^{(\mathrm{dec})}_{m-1}) + \mathcal{Z}^{(\mathrm{dec})}_{m-1}\big),\\
\big(\mathcal{H}^{(\mathrm{cross,se})}_{m,\mathrm{dec}}, \mathcal{H}^{(\mathrm{cross,tr})}_{m,\mathrm{dec}}\big) &= \mathrm{SeriesDecomp}\big(\mathrm{FullAtt}_{\leftrightarrow}\big(\mathcal{H}^{(\mathrm{self,se})}_{m,\mathrm{dec}}, \mathcal{H}^{(\mathrm{ff,se})}_{N,\mathrm{enc}}\big) + \mathcal{H}^{(\mathrm{self,se})}_{m,\mathrm{dec}}\big),\\
\big(\mathcal{H}^{(\mathrm{ff,se})}_{m,\mathrm{dec}}, \mathcal{H}^{(\mathrm{ff,tr})}_{m,\mathrm{dec}}\big) &= \mathrm{SeriesDecomp}\big(\mathrm{FF}\big(\mathcal{H}^{(\mathrm{cross,se})}_{m,\mathrm{dec}}\big) + \mathcal{H}^{(\mathrm{cross,se})}_{m,\mathrm{dec}}\big),\\
\mathcal{X}^{(\mathrm{tr})}_{m,\mathrm{dec}} &= \mathcal{X}^{(\mathrm{tr})}_{m-1,\mathrm{dec}} + \mathrm{Proj}\big(\mathcal{H}^{(\mathrm{self,tr})}_{m,\mathrm{dec}} + \mathcal{H}^{(\mathrm{cross,tr})}_{m,\mathrm{dec}} + \mathcal{H}^{(\mathrm{ff,tr})}_{m,\mathrm{dec}}\big),
\end{aligned}$$

где $(\mathcal{Z}^{(\mathrm{dec})}_{\nu})_{\nu=0}^{M-1} := \big(\mathcal{H}_{0,\mathrm{dec}}, \mathcal{H}^{(\mathrm{ff,se})}_{1,\mathrm{dec}}, \dots, \mathcal{H}^{(\mathrm{ff,se})}_{M-1,\mathrm{dec}}\big)$ и $\mathcal{X}^{(\mathrm{tr})}_{0,\mathrm{dec}} = \mathrm{Proj}(\mathcal{X}^{(\mathrm{tr})}_{\mathrm{dec}})$.

Здесь $\mathcal{H}^{(i,j)}_{m,\mathrm{dec}},\ j \in \{\mathrm{se},\mathrm{tr}\},\ i \in \{\mathrm{self},\mathrm{cross},\mathrm{ff}\}$ – сезонные и трендовые компоненты после соответствующих внутренних подпроцессов (однонаправленного self-attention, двунаправленного cross-attention и feed-forward); $\mathcal{H}^{(\mathrm{ff,se})}_{N,\mathrm{enc}}$ – выход энкодера (прошлая сезонная информация); $\mathrm{Proj}(\cdot)$ – линейное преобразование по признаковому измерению (свёртка с ядром размера 3), переводящее суммарную трендовую компоненту текущего слоя в пространство выходных признаков размерности C_{out}; $\mathrm{FAVOR}(\cdot)$ – механизм внимания FAVOR+ [5] (см. раздел 3), $\mathrm{FullAtt}(\cdot)$ – классический «полный» механизм внимания [18] (см. раздел 2.1).

Таким образом, на каждом слое m self-attention и cross-attention работают только с сезонной составляющей, в то время как трендовые компоненты $\mathcal{H}^{(\cdot,\mathrm{tr})}_{m,\mathrm{dec}}$ проецируются и накапливаются. На выходе декодера получаем сезонное представление $\mathcal{H}^{(\mathrm{ff,se})}_{M,\mathrm{dec}}$ и накопленную трендовую часть $\mathcal{X}^{(\mathrm{tr})}_{M,\mathrm{dec}}$, сумма которых даёт итоговый прогноз:

$$\mathcal{H}^{(\mathrm{ff,se})}_{M,\mathrm{dec}}\mathcal{W}^{(\mathrm{se})} + \mathcal{X}^{(\mathrm{tr})}_{M,\mathrm{dec}} \in \mathbb{R}^{(\frac{L_x}{2}+L_y)\times C_{\mathrm{out}}},$$

где $\mathcal{W}^{(\mathrm{se})} \in \mathbb{R}^{d_{\mathrm{model}}\times C_{\mathrm{out}}}$ – матрица проекции в пространство выходных признаков размерности C_{out}.

В совокупности предложенная архитектура объединяет сильные стороны трёх подходов: локальное кодирование паттернов через ConvStem, линейное по длине последовательности глобальное внимание FAVOR+ и явную декомпозицию ряда из Autoformer [19], при этом структура основных гиперпараметров остаётся совместимой с Informer [21].

4 Эксперимент

Наборы данных. Мы оцениваем Convformer и базовые модели на семи стандартных бенчмарках для задачи долгосрочного прогнозирования временных рядов (LSTF): *ETTh1, ETTh2, ECL, Exchange, Illness, Traffic* и *Weather*. Все наборы данных и протокол их использования заимствованы из работ Informer [21] и Autoformer [19].

Кратко опишем каждый датасет. (1) *ETTh1, ETTh2* [21] содержат почасовые измерения нагрузки и температуры масла электрических трансформаторов за два года. Каждый временной шаг описывается несколькими признаками (температура масла и связанные с ней нагрузки). (2) *Electricity (ECL)* [2] содержит почасовое потребление электроэнергии 321 потребителя в период с 2012 по 2014 год. (3) *Exchange* [13] представляет собой ежедневные курсы обмена валют восьми стран по отношению к доллару США в период с 1990 по 2016 год. (4) *Traffic* [3] – почасовые данные Департамента транспорта Калифорнии, описывающие долю занятости дороги, измеренную различными датчиками на шоссе Сан-Франциско-Бэй. (5) *Weather* [4] содержит метеорологические наблюдения, записанные каждые 10 минут в течение 2020 года (21 показатель, включая температуру воздуха, влажность и

[2] `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`

[3] `http://pems.dot.ca.gov`

[4] `https://www.bgc-jena.mpg.de/wetter/`

др.). (6) *Illness* [5] включает еженедельные значения доли пациентов с симптомами гриппоподобного заболевания (ILI) в США в период с 2002 по 2021 год.

Следуя принятому в Autoformer [19] протоколу, каждую последовательность хронологически делим на обучающую, валидационную и тестовую части: для *ETT (ETTh1, ETTh2)* – 12/4/4 месяцев (6:2:2), для остальных наборов данных – разбиение 7:1:2 по числу временных шагов.

Метрики и протокол оценки. Во всех экспериментах решается задача многомерного прогнозирования: по окну из $L_x = 96$ последних наблюдений модель предсказывает вектор будущих значений длины L_y для выбранных таргетных компонент. Для большинства датасетов рассматриваются горизонты $L_y \in \{24, 48, 168, 336, 720\}$, для Illness – $L_y \in \{24, 36, 48, 60\}$ (см. табл. 5).

Качество оценивается по среднеквадратичной ошибке (MSE) и средней абсолютной ошибке (MAE), считаемым по всему прогнозному окну и всем таргетным компонентам и усреднённым по объектам тестовой выборки. Все числовые признаки проходят z-нормализацию по статистикам обучающей выборки; перед вычислением метрик предсказания денормализуются в исходный масштаб.

Декодер следует схеме Informer [21]: на вход подаётся конкатенация истинного префикса таргетного ряда длины $L_{\text{label}} = L_x/2 = 48$ и L_y стартовых токенов. Значения на горизонте L_y всегда предсказываются за один проход (non-autoregressive, без teacher forcing).

Базовые модели. Convformer сравнивается с Informer [21], Performer [5] и Autoformer [19] в единых условиях. Все модели обучаются и оцениваются в единой постановке задачи LSTF с единым протоколом подготовки данных и общим набором базовых архитектурных гиперпараметров (включая d_{model}, число голов внимания, длины окон, число слоёв энкодера/декодера), взятых из официальных скриптов Autoformer. Специфические параметры внимания (например, число случайных признаков) фиксированы и не зависят от датасета. Таким образом, различия в результатах обусловлены именно архитектурой моделей, а не настройкой задачи или гиперпараметров.

Детали реализации. Обучение проводилось с использованием функции потерь MSE (L2) и оптимизатора Adam [12] с начальной скоростью обучения 10^{-4}. Размер батча равен 32. Обучение досрочно останавливается при отсутствии улучшения валидационной ошибки в течение 10 эпох. Все результаты экспериментов были получены в ходе усреднения по трем отдельным запускам.

Модели реализованы в PyTorch [15] и обучались на одной видеокарте NVIDIA GTX 1660 SUPER с 6 ГБ видеопамяти. Во всех трансформерных архитектурах (Convformer, Informer, Performer, Autoformer) используется одинаковая глубина: 2 слоя энкодера и 1 слой декодера. Полный перечень гиперпараметров и конфигураций приведён в репозитории: `https://github.com/namenick91/Convformer`.

Horizon		**Convformer**			Informer			Performer			Autoformer		
		MSE	MAE	t	MSE	MAE	t	MSE	MAE	t	MSE	MAE	t
ETTh1	24	**0.388**	0.428	3m50s/4s	0.524	0.527	**2m5s/3s**	0.598	0.570	2m50s/4s	0.401	**0.425**	3m58s/7s
	48	0.435	0.451	3m55s/4s	0.631	0.601	**2m24s/4s**	0.765	0.673	2m35s/4s	**0.430**	**0.445**	4m43s/7s
	168	**0.435**	**0.459**	8m39s/6s	0.825	0.705	**3m14s/5s**	0.918	0.768	3m42s/6s	0.478	0.473	6m45s/11s
	336	**0.469**	**0.490**	7m55s/8s	1.310	0.937	9m19s/6s	1.024	0.823	**5m36s/7s**	0.516	0.497	10m19s/17s
	720	**0.510**	**0.528**	**12m51s/10s**	1.205	0.879	14m36s/9s	1.107	0.843	14m9s/9s	–	–	–
ETTh2	24	**0.248**	**0.345**	2m38s/4s	1.284	0.891	**1m58s/3s**	1.296	0.907	2m49s/4s	0.290	0.365	4m7s/6s
	48	**0.298**	**0.373**	2m49s/5s	1.559	1.008	**1m58s/4s**	1.568	1.008	2m35s/4s	0.324	0.382	3m31s/6s
	168	0.539	0.509	4m6s/6s	7.587	2.335	**2m49s/5s**	8.487	2.569	3m40s/6s	**0.451**	**0.456**	5m15s/9s
	336	0.684	0.600	7m9s/8s	4.369	1.773	**4m19s/7s**	8.158	2.456	6m16s/7s	**0.478**	**0.480**	7m3s/13s
	720	**0.662**	**0.595**	13m1s/11s	2.977	1.467	**6m1s/9s**	3.707	1.640	7m39s/9s	–	–	–

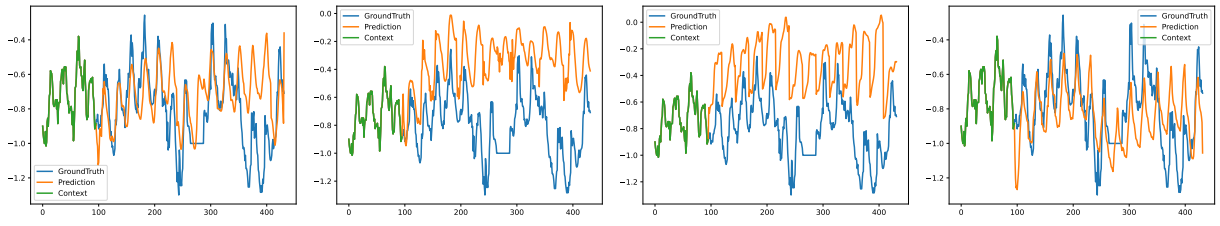
Таблица 1: Результаты многомерных предсказаний на 2 датасетах ETT с горизонтами предсказаний: { 24, 48, 168, 336, 720 }. Мы фиксируем входную длину последовательностей у моделей как 96. Символ "–" обозначает выход за пределы памяти (OOM). Время указано как train/inference. Полный бенчмарк см. в табл. 5. Примеры визуализации приведены на рис. 4.1.

[5] `https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html`

Основные результаты. На ETTh1/ETTh2 (табл. 1) Convformer устойчиво улучшает MSE/MAE по сравнению с Informer и Performer на всех горизонтах; относительно Autoformer он выигрывает на большинстве горизонтов ETTh1, тогда как на ETTh2 Autoformer остаётся лучше на самых длинных (168 и 336).

Полный бенчмарк по семи датасетам (табл. 5) показывает преимущество Convformer на большинстве наборов данных (ETTh1/ETTh2, ECL, Illness, Traffic, Weather), тогда как Autoformer остаётся сильнее главным образом на Exchange и отдельных длинных горизонтах. В суммарной строке *Count* Convformer демонстрирует большее число побед по метрикам по сравнению с базовыми моделями, что указывает на устойчивый прирост качества при сопоставимой вычислительной стоимости.

Визуализация основных результатов Для качественной оценки прогноза различных моделей на рис. 4.1 показана последняя компонента временного ряда из тестового подмножества набора данных ETTh1 для горизонта 336 шагов. Видно, что Convformer и Autoformer дают очень похожие по качеству прогнозы, корректно улавливающие периодичность и долгосрочные изменения сигнала.



Convformer Informer Performer Autoformer

Рис. 4.1: Примеры прогнозирования из набора данных ETTh1 в настройках `input-96-predict-336`. Синие линии – истинные значения, оранжевые линии – предсказания модели, зеленые – входные данные длиной 96.

4.1 Абляционные исследования

ConvStem вместо TokenEmbedding Сравнивалась эффективность оригинальной модели Informer [21] и её модификации с расширенным слоем эмбеддинга (табл. 2).

Horizon	**ConvStem**			Informer		
	MSE	MAE	t	MSE	MAE	t
24	**0.469**	**0.480**	2m11s/4s	0.524	0.527	**2m5s/3s**
48	**0.587**	**0.558**	**2m23s/4s**	0.631	0.601	2m24s/4s
168	0.861	0.733	3m34s/5s	**0.825**	**0.705**	**3m14s/5s**
336	**1.094**	**0.843**	**7m22s/7s**	1.310	0.937	9m19s/6s
720	1.261	0.913	**9m58s/9s**	**1.205**	**0.879**	14m36s/9s

Таблица 2: Результаты многомерных предсказаний на датасете ETTh1 с горизонтами предсказаний: { 24, 48, 168, 336, 720 }. Входная длина последовательности фиксирована: 96. Время указано как train/inference. Примеры визуализации приведены в табл. 6.

Встраивание компактного сверточного блока на вход приводит к заметному улучшению качества на коротких (24-48) и особенно длинных горизонтах (336), при этом затраты по времени остаются сопоставимыми. Однако на отдельных горизонтах (например, 168) Informer [21] сохраняет преимущество. Это подтверждает, что сверточная фильтрация локальных паттернов в среднем повышает устойчивость к краткосрочной нестационарности, хотя её вклад не универсален.

ProbSparse → FAVOR+ Сравнивалась эффективность Informer [21] с исходным механизмом ProbSparse и модифицированной версии, в которой ProbSparse заменён на FAVOR+ (Performer [5]).

Линейное внимание обеспечивает выигрыш по качеству на длинных горизонтах (336, 720) при сопоставимых или меньших вычислительных затратах, в то время как на коротких горизонтах улучшение выражено слабее

Horizon	Informer			**Informer w/ FAVOR+**		
	MSE	MAE	t	MSE	MAE	t
24	0.524	0.527	**2m5s/3s**	**0.494**	**0.505**	2m30s/4s
48	**0.631**	**0.601**	2m24s/4s	0.710	0.640	**2m20s/4s**
168	**0.825**	**0.705**	**3m14s/5s**	0.864	0.746	3m24s/5s
336	1.310	0.937	9m19s/6s	**1.088**	**0.846**	**6m22s/7s**
720	1.205	0.879	14m36s/9s	**1.065**	**0.831**	**9m54s/9s**

Таблица 3: Результаты многомерных предсказаний на датасете ETTh1 с горизонтами предсказаний: { 24, 48, 168, 336, 720 }. Входная длина последовательности фиксирована: 96. Время указано как train/inference. Примеры визуализации приведены в табл. 7.

или отсутствует. Таким образом, FAVOR+ [5] даёт наибольший эффект именно в режимах, где глобальные зависимости становятся критичными, подтверждая его ценность для масштабируемого долгосрочного прогнозирования.

Informer с модулем декомпозиции Сравнивалась эффективность оригинальной модели Informer [21] и модификации, дополненной механизмом декомпозиции ряда из Autoformer [19].

Horizon		Informer			**Informer w/ s.decomp**		
		MSE	MAE	t	MSE	MAE	t
ETTh1	24	0.524	0.527	**2m5s/3s**	**0.478**	**0.494**	3m36s/4s
	48	0.631	0.601	**2m24s/4s**	**0.561**	**0.547**	3m4s/4s
	168	0.825	0.705	**3m14s/5s**	**0.649**	**0.597**	7m2s/5s
	336	1.310	0.937	**9m19s/6s**	**0.952**	**0.748**	10m55s/7s
	720	1.205	0.879	**14m36s/9s**	**1.059**	**0.775**	16m46s/10s

Таблица 4: Результаты многомерных предсказаний на датасете ETTh1 с горизонтами предсказаний: { 24, 48, 168, 336, 720 }. Входная длина последовательности фиксирована: 96. Время указано как train/inference. Примеры визуализации приведены в табл. 8.

Встраивание операции разделения на тренд и остаток после каждого блока self-attention приводит к систематическому снижению ошибок на всех горизонтах ETTh1, хотя ценой становится рост времени обучения и инференса (inference). Это подтверждает гипотезу о том, что явная стабилизация нестационарности улучшает обобщающую способность модели, особенно при наличии сдвигов уровня и мультисезонности.

5 Заключение

В работе предложен Convformer – модульная модификация Informer, объединяющая сверточный входной блок ConvStem, линейное внимание FAVOR+ и Autoformer-подобную декомпозицию временного ряда. Такая комбинация индуктивных смещений позволяет более явно разделить задачи выделения локальных паттернов, моделирования дальних зависимостей и компенсации нестационарности.

На семи стандартных бенчмарках LSTF (ETTh1/ETTh2, ECL, Exchange, Illness, Traffic, Weather) Convformer обеспечивает снижение ошибок прогноза на большинстве горизонтов по сравнению с Informer и Performer при сопоставимых вычислительных затратах. Относительно Autoformer выигрыш достигается на большинстве датасетов, за исключением Exchange и отдельных длинных горизонтов, где Autoformer остаётся конкурентным. Абляционные исследования показывают, что каждый из трёх модулей вносит вклад в качество: ConvStem улучшает обработку краткосрочных мотивов и устойчивость к локальным скачкам, FAVOR+ даёт масштабируемый учёт глобальных связей на длинных горизонтах, декомпозиция повышает устойчивость к тренду и мультисезонности. На ETT наибольшие выигрыши достигаются при длительных горизонтах (336–720), тогда как на средних горизонтах Autoformer остаётся сопоставимым. Абляционные исследования и подробный бенчмарк приведены в разделах 4, 4.1 и приложении A.

При этом исследование имеет ряд ограничений: cross-attention сохраняет квадратичную сложность; настройки модели и гиперпараметров подбирались в ограниченном классе конфигураций; не проводилось систематическое исследование глубины, ширины и размеров карт случайных признаков; отсутствие оценки неопределённости. Перспективными направлениями являются линеаризация cross-attention и/или внедрение модуля дистилляции в декодер, адаптивный выбор ранга r в FAVOR+, а также включение калибровки и методов, устойчивых к дрейфу распределения данных. Отдельная линия работы – перенос предложенной модульной схемы на другие архитектуры блоков энкодера/декодера и задачи за пределами LSTF.

Список использованных источников

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations, 2015. https://arxiv.org/abs/1409.0473.

[2] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. Time Series Analysis: Forecasting and Control. John Wiley & Sons, fifth edition edition, 2015.

[3] Peter J. Brockwell and Richard A. Davis. Introduction to Time Series and Forecasting. Springer, third edition edition, 2016.

[4] Krzysztof Choromanski, Haoxian Chen, Han Lin, Yuanzhe Ma, Arijit Sehanobish, Deepali Jain, Michael S. Ryoo, Jake Varley, Andy Zeng, Valerii Likhosherstov, Dmitry Kalashnikov, Vikas Sindhwani, and Adrian Weller. Hybrid random features. In International Conference on Learning Representations, 2022. Conference paper at ICLR 2022. https://arxiv.org/abs/2110.04367.

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Q. Davis, Afroz Mohiuddin, Łukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In International Conference on Learning Representations, 2021. Oral presentation.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. https://aclanthology.org/N19-1423/.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[8] James D. Hamilton. Time Series Analysis. Princeton University Press, 1994.

[9] Rob J. Hyndman and George Athanasopoulos. Forecasting: Principles and Practice. OTexts, Melbourne, Australia, 2013. https://otexts.com/fpp/.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 448–456, Lille, France, 2015. PMLR. https://proceedings.mlr.press/v37/ioffe15.html.

[11] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. fast-transformers: PyTorch library for fast transformer implementations. https://github.com/idiap/fast-transformers, 2020. Release version 0.3.0, MIT License.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), 2015. Poster presentation. https://arxiv.org/abs/1412.6980.

[13] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 95–104, Ann Arbor, MI, USA, 2018. ACM. https://doi.org/10.1145/3209978.3210006.

[14] Kevin P. Murphy. Probabilistic Machine Learning: An Introduction. MIT Press, 2022. http://probml.github.io/book1.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, volume 32, pages 8024–8035. Curran Associates, Inc., 2019. https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[16] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In Advances in Neural Information Processing Systems, volume 20, pages 1177–1184. Curran Associates, Inc., 2007. https://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016. https://arxiv.org/abs/1607.08022.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30, pages 5998–6008. Curran Associates, Inc., 2017. https://papers.nips.cc/paper/7181-attention-is-all-you-need.

[19] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Advances in Neural Information Processing Systems, volume 34, pages 22419–22430. Curran Associates, Inc., 2021.

[20] Haoyi Zhou, Jianxin Li, Shanghang Zhang, Shuai Zhang, Mengyi Yan, and Hui Xiong. Expanding the prediction capacity in long sequence time-series forecasting. Artificial Intelligence, 318:103886, 2023. https://doi.org/10.1016/j.artint.2022.103886.

[21] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference, volume 35, pages 11106–11115. AAAI Press, 2021. https://ojs.aaai.org/index.php/AAAI/article/view/17325.

Приложение A Полный бенчмарк

Models		**Convformer**		Informer		Performer		Autoformer	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	**0.388**	0.428	0.524	0.527	0.598	0.570	0.401	**0.425**
	48	0.435	0.451	0.631	0.601	0.765	0.673	**0.430**	**0.445**
	168	**0.435**	**0.459**	0.825	0.705	0.918	0.768	0.478	0.473
	336	**0.469**	**0.490**	1.310	0.937	1.024	0.823	0.516	0.497
	720	**0.510**	**0.528**	1.205	0.879	1.107	0.843	–	–
ETTh2	24	**0.248**	**0.345**	1.284	0.891	1.296	0.907	0.290	0.365
	48	**0.298**	**0.373**	1.559	1.008	1.568	1.008	0.324	0.382
	168	0.539	0.509	7.587	2.335	8.487	2.569	**0.451**	**0.456**
	336	0.684	0.600	4.369	1.773	8.158	2.456	**0.478**	**0.480**
	720	**0.662**	**0.595**	2.977	1.467	3.707	1.640	–	–
ECL	24	**0.161**	**0.282**	0.253	0.356	0.265	0.371	0.170	0.290
	48	**0.176**	**0.293**	0.273	0.366	0.276	0.373	0.182	0.298
	168	**0.198**	**0.312**	0.288	0.381	0.286	0.383	0.220	0.329
	336	**0.217**	**0.330**	0.315	0.404	0.301	0.394	0.261	0.359
	720	–	–	–	–	–	–	–	–
Exchange	24	0.078	0.213	0.420	0.523	0.272	0.418	**0.065**	**0.185**
	48	**0.118**	0.257	0.584	0.614	0.437	0.520	0.119	**0.253**
	168	0.493	0.501	1.142	0.873	1.352	0.904	**0.267**	**0.378**
	336	0.621	0.581	1.767	1.061	1.892	1.071	**0.468**	**0.508**
	720	**0.591**	**0.597**	2.834	1.404	2.783	1.405	–	–
Illness	24	**3.021**	**1.158**	5.451	1.615	5.130	1.566	3.645	1.353
	36	**2.962**	**1.147**	5.059	1.541	4.904	1.502	3.511	1.300
	48	**3.040**	**1.145**	5.028	1.536	4.733	1.445	3.149	1.197
	60	2.921	**1.113**	5.340	1.578	5.212	1.533	**2.905**	1.152
Traffic	24	**0.533**	**0.341**	0.705	0.398	0.667	0.382	0.587	0.388
	48	**0.556**	**0.352**	0.685	0.380	0.666	0.373	0.606	0.385
	168	**0.597**	0.378	0.740	0.404	0.669	**0.368**	–	–
	336	**0.620**	0.386	0.770	0.420	0.678	**0.372**	–	–
Weather	24	**0.124**	**0.204**	0.155	0.237	0.170	0.255	0.164	0.245
	48	**0.167**	**0.248**	0.240	0.323	0.320	0.392	0.233	0.309
	168	**0.271**	**0.347**	0.490	0.497	0.538	0.521	0.294	0.354
	336	0.425	0.460	0.626	0.554	0.657	0.577	**0.354**	**0.389**
	720	**0.652**	**0.596**	0.912	0.687	0.925	0.701	–	–
Count		45		0		2		17	

Таблица 5: Результаты многомерного прогнозирования на семи датасетах (ETTh1, ETTh2, ECL, Exchange, Illness, Traffic, Weather) при различных горизонтах предсказаний $\{24, 48, 168, 336, 720\}$ (для Illness: $\{24, 36, 48, 60\}$). Входная длина последовательности фиксирована: 96. Число прогнозируемых признаков зависит от датасета. Наилучшие результаты по каждому датасету, горизонту и метрике выделены полужирным. Символ «–» обозначает выход за пределы памяти (OOM).

Приложение B Дополнение к основным результатам

Визуализация абляционных исследований Для качественной оценки моделей представлена визуализация прогнозов последней компоненты временного ряда набора данных *ETTh1* [21] на тестовой выборке. Показаны

результаты для нескольких горизонтов прогнозирования в рамках абляционных исследований, что позволяет наглядно сравнить характер ошибок и способность моделей отслеживать динамику ряда.

Таблица 6: Примеры прогнозирования из набора данных ETTh1 в ходе абляционных исследований *ConvStem вместо TokenEmbedding* (см. раздел 4.1) в настройках `input-96-predict-horizon`. Синие линии – истинные значения, оранжевые линии – предсказания модели, зеленые – входные данные длиной 96.

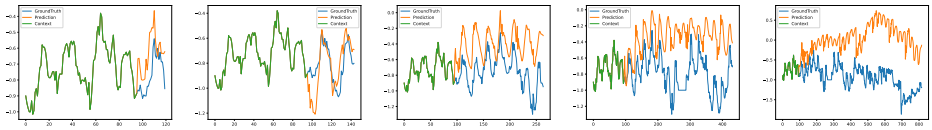
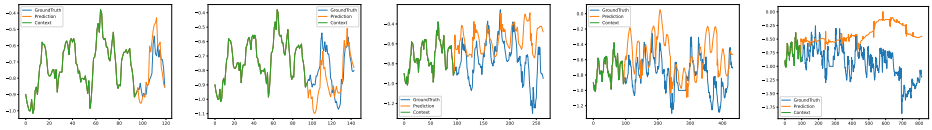
Model	Horizon				
	24	48	168	336	720
Informer					
ConvStem					

Таблица 7: Примеры прогнозирования из набора данных ETTh1 в ходе абляционных исследований *ProbSparse* $\rightarrow$ *FAVOR+* (см. раздел 4.1) в настройках `input-96-predict-horizon`.

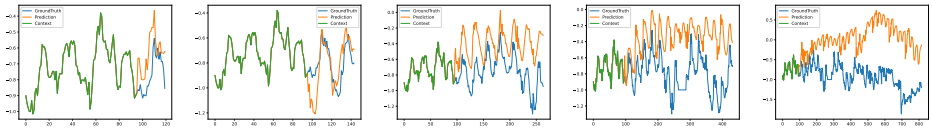
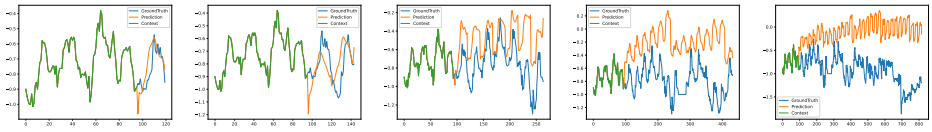
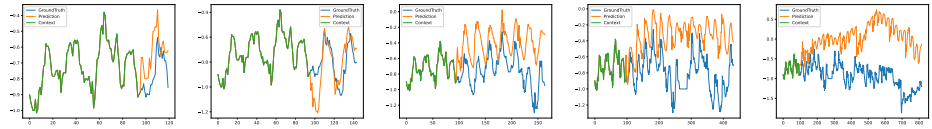
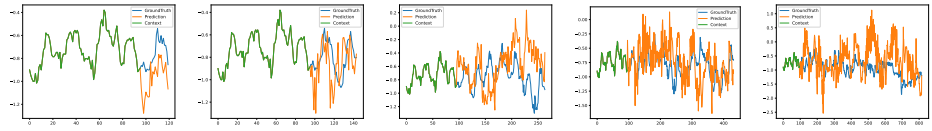
Model	Horizon				
	24	48	168	336	720
Informer					
w/ FAVOR+					

Таблица 8: Примеры прогнозирования из набора данных ETTh1 в ходе абляционных исследований *Informer с модулем декомпозиции* (см. раздел 4.1) в настройках `input-96-predict-horizon`.

Model	Horizon				
	24	48	168	336	720
Informer					
w/ s.decomp					

Приложение C Детали реализации

В настоящем разделе фиксируем ключевые детали реализации Convformer.

Algorithm 1 Блок ConvStem

Require: входное окно $\mathcal{X}_t \in \mathbb{R}^{B \times L_x \times C_{\text{in}}}$
Ensure: выход $\mathcal{X}'_t \in \mathbb{R}^{B \times L_x \times d_{\text{model}}}$

$\mathcal{X}_t \leftarrow \texttt{Permute}(\mathcal{X}_t)$ ▷ $\mathcal{X}_t \in \mathbb{R}^{B \times C_{\text{in}} \times L_x}$
$R \leftarrow \texttt{Conv1D}_{k=1,p=0}(\mathcal{X}_t)$ ▷ $R \in \mathbb{R}^{B \times d_{\text{model}} \times L_x}$
$H_1 \leftarrow \texttt{GELU}\big(\texttt{IN}(\texttt{Conv1D}_{k=5,p=2}(\mathcal{X}_t))\big)$ ▷ $H_1 \in \mathbb{R}^{B \times d_{\text{model}} \times L_x}$
$H_2 \leftarrow \texttt{GELU}\big(\texttt{IN}(\texttt{DW-Conv1D}_{k=3,p=1}(H_1))\big)$ ▷ $H_2 \in \mathbb{R}^{B \times d_{\text{model}} \times L_x}$
$Z \leftarrow R + H_2$ ▷ $Z \in \mathbb{R}^{B \times d_{\text{model}} \times L_x}$
$\mathcal{X}'_t \leftarrow \texttt{Permute}^{-1}(Z)$ ▷ $\mathcal{X}'_t \in \mathbb{R}^{B \times L_x \times d_{\text{model}}}$
return $\mathcal{X}'_t$

Комментарий. Здесь `IN` обозначает слой InstanceNorm1d [17] с обучаемыми параметрами (affine = True), `DW-Conv1D` – глубинную одномерную свёртку по временной оси (groups $= d_{\text{model}}$), а GELU – функцию активации.

Encoder:		N
Inputs	ConvEmbedding ($d_{\text{model}} = 512$)	2
FAVOR+ self-attention block & decomposition	Multi-head FAVOR+ attention ($h = 8$, $d_k = 64$, $r = 256$) Add, SeriesDecomp ($k = 25$), Dropout ($p = 0.05$) Pos-wise FFN ($d_{ff} = 2048$), RELU Add, SeriesDecomp ($k = 25$), Dropout ($p = 0.05$)	
Distilling	Conv1d ($k = 3, p = 2$), BatchNorm1d ELU, MaxPool1d ($k = 3, p = 1, s = 2$)	
Decoder:		M
Inputs	ConvEmbedding ($d_{\text{model}} = 512$)	1
FAVOR+ self-attention block	Multi-head FAVOR+ attention ($h = 8, d_k = 64, r = 256$) + Mask	
Full Attention cross-attention block & decomposition	Multi-head Attention ($h = 8, d_k = 64$) Add, SeriesDecomp ($k = 25$), Dropout ($p = 0.05$) Pos-wise FFN ($d_{ff} = 2048$), RELU Add, SeriesDecomp ($k = 25$), Dropout ($p = 0.05$)	
Final:		
Outputs	Seasonal part + trend part, Linear ($d_{\text{model}} \to C_{\text{out}}$)	

Таблица 9: Гиперпараметры и конфигурация архитектуры предлагаемой модели (Convformer).