ChatGPT

Модель N-BEATS для прогнозирования временных рядов: технический обзор

1. Архитектура N-BEATS (2020): идеи и компоненты

Общая идея. *Neural Basis Expansion Analysis for Time Series* (N-BEATS) – это нейросетевая архитектура для одномерного прогноза временных рядов, впервые представленная Oreshkin et al. в 2019/2020 годах [1]. Ключевая особенность – полносвязные блоки с **двойными остаточными связями** (backward и forward residual links), образующие глубокую каскадную структуру [2]. Модель намеренно *не* использует рекуррентных или свёрточных слоёв – авторы показывают, что одни лишь стандартные блоки MLP с residual-архитектурой могут достичь SOTA-точности на разнородных задачах прогнозирования [3], опровергая представление о необходимости специализированных архитектур для рядов.

Блоки, backcast/forecast. ***В архитектуре N-BEATS входной временной ряд длины T (lookback window) последовательно проходит через* стек из L блоков*. Каждый блок l выдаёт два выхода: (1)** backcast **$x^{(l)}$ – приближение (реконструкцию) входного фрагмента ряда; (2)** forecast *$y^{(l)}$ – частичный прогноз на горизонт H вперед. Внутри блока находится несколько (обычно 4) полносвязных слоёв ReLU, по окончании которых сеть разветвляется на две "головы", генерируя параметры θ_l^b и θ_l^f для backcast и forecast соответственно [4]. Эти параметры интерпретируются как коэффициенты разложения по некоторому базису. Далее через линейные преобразования они разворачиваются в сам backcast-выход блока (размерности T) и прогноз блока на H шагов [5]. Таким образом блок реализует базисное разложение*: $x^{(l)} = B(\theta_l^b)$, $y^{(l)} = F(\theta_l^f)$, где $B(\cdot)$ и $F(\cdot)$ – некоторые функции (или матрицы) формирования сигнала из коэффициентов. В простейшем случае (generic) эти базисы тождественные – коэффициенты напрямую являются значениями точек ряда (identity basis). В специальных конфигурациях B,F могут задаваться априорно выбранными семействами функций (например, полиномы, гармоники) – об этом ниже.

Residual stacking (двойная резидуальная связь). Термин «двойной residual» отражает два уровня остаточных связей в архитектуре [2]. Во-первых, **backward residual**: выход backcast текущего блока вычитается из входного сигнала блока, и **остаток** (невырезанная часть временного ряда) передаётся на вход следующему блоку [6]. Это аналог подхода бустинга: каждый следующий блок моделирует то, что не объяснили предыдущие. Во-вторых, **forward residual**: частичные прогнозы $y^{(l)}$ от всех блоков суммируются (или аггрегируются) для формирования итогового прогноза модели [7]. Таким образом, модель складывает вклад каждого блока в конечный прогноз, обеспечивая дополнительную прямую связь градиентов (skip connection) от каждого блока к выходу. Такой дизайн стабилизирует обучение очень глубокой сети – суммарно N-BEATS может иметь десятки блоков по 4 слоя (итоговая глубина ~120 слоёв), однако residual-связи облегчают оптимизацию.

Generic vs Interpretable конфигурации. Авторами предложено две основные конфигурации архитектуры [8] :

- *Generic (общая)* – **не** содержит специфических для временных рядов компонентов. Все блоки используют *тождественный базис* (identity), т.е. каждый блок просто обучается произвольно преобразовывать вход и выдавать нужный сигнал. Для повышения мощности модели в generic-режиме обычно используют большее число блоков и параметров. Необычно, но такой свободный MLP-подход показал себя исключительно хорошо на разнородных данных, что указывает: даже без явного кодирования сезонности/тренда сеть способна их выучить [3] . Generic-конфигурация дала SOTA-результаты на конкурсах M3, M4 и Tourism, превзойдя предыдущего победителя M4 (модель ESRNN) ~на 3% по точности [1] .

- *Interpretable (интерпретируемая)* – включает *специализированные стеки* блоков для улавливания тренда и сезонности. Здесь архитектура разбивается на два (или несколько) стеков: например, один стек из блоков типа **Trend**, другой – типа **Seasonality** [9] . В блоках тренда выходные коэффициенты θ^b, θ^f интерпретируются как коэффициенты при базисе полиномиальных функций времени (обычно полиномы невысокого порядка), а в блоках сезонности – при базисе гармонических функций (набор синусов/косинусов заданных частот) [10] . Благодаря этому *частичные прогнозы* $y^{(trend)}$, $y^{(season)}$ легко суммируются в итоговый прогноз и прямо представляют трендовую и сезонную компоненты, что повышает объяснимость. В оригинальной работе интерпретируемая модель использовала 2 стека (тренд и сезон) по 3 блока в каждом [11] , с общим числом параметров меньше, чем у generic-аналога. Несмотря на ограничение формы выходов, такая модель практически не уступает по точности: авторы отметили “минимальную потерю точности” ради интерпретируемости [12] . Более того, интерпретируемый N-BEATS служит регуляризатором при малых данных – навязывание априорных форм (гармоники/полиномы) не даёт переобучиться на шуме и показано улучшает результат в условиях дефицита истории [13] .

Формализация блока. Пусть $x \in \mathbb{R}^{T}$ – вход блока (исходный ряд для первого блока или остаток после предыдущего). Блок состоит из нескольких FC-слоёв: $h = \mathrm{ReLU}(W_4 \mathrm{ReLU}(W_3 \mathrm{ReLU}(W_2 \mathrm{ReLU}(W_1 x))))$ (например, 4 слоя по 512 нейронов). Затем последним линейным слоем раздельно формируются вектора параметров $\theta^b \in \mathbb{R}^{P}$ и $\theta^f \in \mathbb{R}^{Q}$ – для backcast и forecast соответственно [4] . **Базисные функции** задаются матрицами фиксированных весов: G^b размером $T\times P$ и G^f размером $H\times Q$. Тогда выходы блока:

$$\hat{x} = G^b\,\theta^b, \quad \hat{y} = G^f\,\theta^f,$$

где $\hat{x}$ – реконструкция части ряда (backcast), $\hat{y}$ – прогноз блока. В generic-режиме G^b и G^f обычно берутся как единичные матрицы (т.е. $P=T$, $Q=H$), фактически блок напрямую генерирует значения временных точек. В interpretable-режиме для **Trend** берут G^f_{trend} как матрицу Вандермонда $[t^0, t^1, \dots, t^k]$ (полиномиальный базис степени k; k=2 или 3) на интервале прогноза, а G^b_{trend} – аналогично на интервале входа. Для **Seasonality** G^f_{season} составляется из пар ${\sin(2\pi jt / \omega), \cos(2\pi jt/\omega)}$ с разными периодами ω (например, гармоники до порядка K), покрывающими нужные циклы; G^b_{season} – те же гармоники на окне входа [14] . Коэффициенты θ^b,θ^f в этих случаях обучаются трактоваться как веса при заданных базисных функциях. Такая параметризация навязывает **структурную интерпретацию**: например, блок-тренд всегда выдаёт сглаженную полиномиальную кривую, блок-сезонность –

комбинацию синусоид, которую можно разложить по частотам. В совокупности суммарный прогноз N-BEATS в интерпретируемой конфигурации представляет собой разложение по трендовой и сезонной компонентам, сравнимое с классическими моделями (напр. сумма полинома и Fourier-составляющих), но с тем отличием, что коэффициенты и сами базисы частично обучаются под данные.

Обучение и метрики. N-BEATS обучается методом градиентного спуска как стандартная нейросеть. В оригинале особое внимание уделялось выбору функции потерь: для достижения наилучших результатов авторы обучали ансамбль из моделей, оптимизирующих разные метрики (sMAPE, MAPE, MASE) [15]. Кроме того, использовались разные длины входного окна (*context length*) – от 2 до 6 горизонтов прогноза – и усреднение (бэггинг) по моделям с разными инициализациями [16] [17]. Финальный ансамбль, состоящий до 180 моделей, выдавал медианный прогноз по их выходам [18]. Такой подход позволил существенно повысить устойчивость и точность (на M4 ансамбль дал ~11% выигрыш над наивным статистическим ориентиром) [1]. В практических библиотеках (например GluonTS) эта стратегия реализована через класс NBEATSEnsembleEstimator [19], где задаётся список длины окон и метрик, по которым тренируются подмодели, и затем их агрегирование (медиана/среднее) [20] [21]. Стоит отметить, что в отличие от многих других моделей GluonTS, N-BEATS в этой реализации не использует вероятностного выхода (распределения) – он оптимизируется **непосредственно по точечным метрикам**, таким как sMAPE/MASE [22]. Вероятностный прогноз можно получить апостериори, оценивая разброс ансамбля или используя дропаута.

Итоги по N-BEATS. Впервые в соревновании по прогнозированию удалось добиться, чтобы *чисто нейросетевой метод превзошёл статистические и гибридные* – N-BEATS на разнородных бенчмарках (M3, M4, Tourism) улучшил точность на ~3% относительно лучшей традиционной комбинации (ESRNN) [1]. При этом архитектура универсальна (не требует специфичного препроцессинга для разных доменов), *быстро обучается* (за счёт MLP-параллелизма и отсутствия рекуррентности) [23] и может быть сделана интерпретируемой без значимой потери качества [12]. Это открыло путь к дальнейшему развитию модели для более сложных сценариев – в следующих разделах рассмотрены расширения N-BEATS, добавляющие поддержку экзогенных факторов и улучшающие работу на длинных горизонтах.

2. Расширения: N-BEATSx и N-HiTS

N-BEATSx: учёт экзогенных факторов (2021)

Мотивация. Оригинальный N-BEATS рассчитан на *унитивариатный* прогноз, используя только историю самого ряда. В реальных задачах часто доступны **экзогенные переменные** – связанные временные сигналы (праздники, экономические индикаторы, прогноз погоды, нагрузки и т.п.), а также *статические свойства* серии (категория товара, регион и др.). Отсутствие механизма интеграции таких ковариат ограничивает применение N-BEATS – например, в конкурсе M5 (продажи Walmart) модели требовалось учитывать промо-акции, цены, календарь; или в прогнозе цен на электроэнергию – учёт нагрузки, генерации ВИЭ и т.д. [24] [25]. Чтобы преодолеть эти ограничения, Olivares et al. (2021) предложили расширение **N-BEATSx** – *Neural Basis Expansion with Exogenous variables*, добавив специальный модуль для обработки экзогенных факторов [26].

Архитектура. N-BEATSx сохраняет общую структуру стека блоков N-BEATS, включая вариантивность generic/interpretable. Основное новшество – параллельный *экзогенный канал*. В модель вводятся: (a) **временные экзогенные регрессоры** (динамические, зависящие от времени, разделяемые на исторические – известные на интервале входа, и на будущие –

известные вплоть до горизонта прогноза); (b) **статические экзогенные признаки** (константные для конкретного ряда, например ID категории). В N-BEATSx реализован отдельный *энкодер экзогенных сигналов* на основе свёрточной сети. Авторы отмечают, что выбрали **Temporal Convolutional Network (TCN)** как подслой для экзогенных, поскольку свёртки с пропускной связностью хорошо вычленяют нужные временные паттерны и могут моделировать зависимости долгой памяти [27] [28]. В общем случае вместо TCN можно применить любую архитектуру для последовательностей (например, WaveNet), совместимую по размерности с выходами backcast/forecast блока [29]. Конкретно, экзогенный энкодер преобразует ковариаты во внутреннее представление (фильтрует и агрегирует информацию во времени) и передаёт её в блоки N-BEATS.

В интерпретируемой версии N-BEATSx экзогенные переменные учитываются **аддитивно** наряду с трендом и сезонностью. Фактически появляется третий тип выходной компоненты – *экзогенная* – наряду с трендовой и сезонной [30] [31]. Блоки экзогенного стека проецируют ковариаты на базисные функции (можно рассматривать, что для экзогенных тоже имеется свой базис – например, набор импульсных функций под каждые регрессоры, либо сглаживающие сплайны) [26] [32]. Таким образом интерпретируемый N-BEATSx способен декомпозировать прогноз на три части: тренд, сезонность и вклад экзогенных факторов [33]. Статические экзогенные признаки учитываются через *общую модель* – N-BEATSx поддерживает обучение **глобальной модели** на множество серий, где статические ковариаты помогают дифференцировать их (например, регион или товар). Статические переменные просто конкатенируются к скрытым представлениям или влияют через condition-блоки, обеспечивая *разделение параметров* по сегментам данных [34] [35].

Результаты. N-BEATSx показал значительные улучшения там, где экзогенные действительно важны. В задаче прогноза цен на электроэнергию (где имеются прогнозы нагрузки и генерации как регрессоры) N-BEATSx превзошёл оригинальный N-BEATS почти на **20% по точности** и обошёл специализированные модели (~5% выигрыш над лучшими статистическими и ML-методами для этой задачи) [25] [36]. При этом интерпретируемость сохранилась: модель позволяет визуализировать вклад тренда/сезонности и каждого экзогенного фактора во временные колебания цены [33]. N-BEATSx был опубликован в 2021 г. (International Journal of Forecasting, 2023) командой из CMU и Вроцлавского политеха, а исходный код выложен в открытый доступ. В экосистеме Nixtla N-BEATSx реализован как часть библиотеки NeuralForecast и отмечен как “официальная реализация” авторов [37].

Вывод: Дополнение N-BEATS экзогенным энкодером расширяет область применения модели на случаи, где важны календарные эффекты, промо-акции, погодные и другие внешние влияния. При этом сохраняются преимущества исходной архитектуры – скорость обучения (сеть всё так же основана на MLP), гибкость (generic/interpretable режимы) и возможность глобального обучения на множестве серий. Ограничение – возросшая сложность модели: свёрточный экзогенный модуль добавляет параметры и требует тюнить его гиперпараметры (например, глубину TCN, окно ковариат). Но выигрыши в точности (двузначные проценты в некоторых задачах) оправдывают эту сложность для практически важных сценариев.

N-HITS: иерархическая интерполяция и мульти-частотность (2022)

Проблема долгосрочного прогноза. Ещё одно направление развития – повышение эффективности на *длинных горизонтах* прогнозирования (десятки, сотни шагов вперёд). При увеличении H у N-BEATS могут возникать две проблемы: (1) *Рост волатильности* прогнозов – модель может генерировать быстрые флуктуации на дальних точках, особенно если ошибка накапливается; (2) *Комбинаторный рост параметров* – для прямого прогноза на H шагов каждый блок должен выдавать вектор длины H (в identity базисе), что увеличивает размер последнего слоя и нагрузку на обучение. В 2022 г. Challu et al. предложили архитектуру **N-HiTS**

(Neural Hierarchical Interpolation for Time Series), нацеленную на решение этих проблем [38]. N-HiTS сохраняет базовый принцип N-BEATS (блоки MLP с backcast/forecast), но вводит два новшества: **иерархическую интерполяцию** и **многочастотную выборку данных**.

Мульти-масштабные блоки. Идея N-HiTS – заставить разные блоки отвечать за разные масштабы частот/длительностей, тем самым снизив суммарную сложность. Для этого в архитектуре применяется механизм **мульти-рейтового pooling'а**: входные данные для разных блоков берутся с разным шагом дискретизации (downsampling). Например, первый блок может оперировать с рядом, усреднённым/сэмплированным через каждый 4-й шаг, второй – через каждый 2-й, а третий – на полной частоте. Соответствующие прогнозы блоков затем **интерполируются** обратно на исходную временную шкалу и суммируются. Такое решение реализовано технически через три элемента:

- **Pooling на входах блоков.** N-HiTS группирует блоки в несколько *стеков*, каждому стеку сопоставляется свой коэффициент downsampling. Например, стек 1 получает на вход исходный ряд с пониженным разрешением (сглаженным ядром размерности 2 или 4) [39]; стек 2 – с меньшим понижением (ядро 2 или без изменения); последний стек – без downsampling (полная детализация). В реализации Nixtla по умолчанию три стека с коэффициентами downsample 4, 2 и 1 (т.е. первый видит сильно сглаженный тренд, последний – все детали) [40].

- **Иерархическая интерполяция прогнозов.** Каждый стек выдаёт свой частичный прогноз – на «своей» временной сетке. Затем прогноз **поднимается** до исходной частоты с помощью заданного метода интерполяции (например, линейной) [40]. Грубо говоря, первый стек прогнозирует общую форму на всём горизонте (например, медленную компоненту), интерполируется до H точек; второй стек прогнозирует более быстрые колебания на промежуточном уровне и дополняет первый; третий стек добавляет самые высокочастотные детали. Сумма этих интерполированных выходов даёт итоговый прогноз.

- **Backcast с учётом масштаба.** Для обратной связи (вычитания остатка) каждый блок также реконструирует свой backcast на соответствующей временной сетке, который аппроксимирует ту часть входного сигнала, за которую блок отвечает. Затем, аналогично N-BEATS, этот backcast вычитается из входа (в нужном масштабе) и передаётся дальше. Благодаря разным масштабам, *остатки вычисляются по полосам частот*: высокочастотные блоки видят только тот остаток, что не объяснили медленные блоки, и наоборот.

Таким образом, N-HiTS добивается специализации блоков по частотным диапазонам. Низкочастотные (сильнопулингованные) блоки моделируют плавную часть ряда, высокочастотные – мелкие колебания [41] [42]. Архитектура по-прежнему MLP-ориентированная, с двойными residual-связями (backward/forward) между блоками [43], поэтому сохраняется простота обучения. Однако за счёт уменьшения размерности входов/выходов отдельных блоков достигается *значительная экономия параметров и вычислений*. Авторы теоретически показали, что иерархическая интерполяция позволяет эффективно аппроксимировать сколь угодно длинные горизонты при условии некоторой гладкости сигналов [44] – т.е. модель масштабируется на длинный прогноз лучше, чем прямая экстраполяция MLP.

Достижения. В экспериментах на наборах для **долгосрочного прогнозирования** (Electricity, Traffic, Weather, ETTh/ETTm и др., горизонты 96, 336, 720 точек) N-HiTS значительно превзошёл современные трансформерные модели. Сообщается улучшение средней точности ~**20–25%**

относительно лучших на тот момент Transformer-архитектур (Informer, Autoformer, ETSformer, PatchTST и др.) [45]. При этом N-HiTS добился *в десятки раз меньшего времени* прогнозирования: так, на длинных рядах он ~50 раз быстрее, чем стандартный Transformer [46] [45]. Такая эффективность обусловлена отсутствием само-attention и меньшим числом параметров (большая часть весов N-HiTS сосредоточена в MLP для локальных моделей). При прочих равных, N-HiTS также показал более стабильный прогноз без излишней волатильности на дальних отрезках, что достигается "сдерживанием" высокочастотных компонентов через многоступенчатую реконструкцию.

Интерпретируемость и применимость. Хотя основной упор в N-HiTS – точность и эффективность, модель по-прежнему позволяет некоторую интерпретацию. Поскольку выход составляется из нескольких частичных прогнозов разной частоты, их можно рассматривать как *декомпозицию по временны́м масштабам*. Это напоминает разложение с помощью вейвлетов или многомасштабной декомпозиции, только в данном случае компоненты учатся автоматически [32]. Прямого экономического смысла эти компоненты не несут (в отличие от тренда/сезонности в N-BEATS), но могут помочь понять, какие частоты доминируют. N-HiTS способен включать экзогенные переменные аналогично N-BEATSx (Nixtla-вариант поддерживает `futr_exog` и пр. в параметрах модели [47]). Однако публикация Challu et al. в AAAI'23 не акцентировала внимание на экзогенных факторах – основной таргет были синтетические наборы без ковариат. Тем не менее, в открытой реализации (NeuralForecast) N-HiTS совместим с теми же экзогенными вводами, что и N-BEATSx, то есть можно применять его для многомерных прогнозов с ковариатами. N-HiTS официально представлен на AAAI 2023 и его открытый код доступен; библиотека NeuralForecast включает *официальную реализацию N-HiTS* от авторов [37].

Вывод: N-HiTS развивает идею N-BEATS, адаптируя её к долгосрочным прогнозам и улучшая масштабируемость. Ввод иерархии масштабов и pooling сделал модель более экономичной и точной на больших горизонтах, что особенно ценно для таких областей, как энергетика (длительный прогноз нагрузки), транспорт, климатические данные и т.п. В сочетании с предыдущим расширением (N-BEATSx) можно получить модель, одновременно учитывающую экзогенные факторы и эффективно обрабатывающую разные временные масштабы – это одно из направлений современных работ.

3. Open-source реализации N-BEATS и её модификаций

С момента публикации N-BEATS получила несколько открытых реализаций. Ниже приведён обзор основных из них – их особенности API, функциональность и различия. Большинство реализаций распространяются под пермиссивными лицензиями (Apache 2.0 или MIT), что упрощает использование в коммерческих проектах.

Официальная реализация (PyTorch, ServiceNow Research)

Авторы N-BEATS выложили исходный код модели на GitHub (репозиторий `ServiceNow/N-BEATS`) [48]. Этот код написан на PyTorch и организован для **воспроизведения экспериментов статьи** [49] [50]. В репозитории присутствуют: модуль `models/nbeats.py` с определением модели, скрипты для загрузки датасетов M3/M4/Tourism, конфигурационные файлы (в формате `.gin`) для различных сценариев (generic vs interpretable, разные длины входа и метрики) [51] [52]. Запуск обучения предполагается через Docker-контейнер и Makefile с преднастроенными командами (пошагово: сборка образа, загрузка данных, генерация конфигов, запуск обучений, сбор результатов) [53] [54]. По умолчанию тренируются ансамбли из 10 моделей на комбинациях 5

разных горизонтов и 3 метрик (итого до 150 моделей) [55] [56] – этот сложный процесс автоматизирован в коде.

API и удобство. Официальный код ориентирован скорее на исследователей, чем на прикладное использование. Он не предоставляет простого класс-метода `fit/predict` для произвольных данных – вместо этого пользователь должен подготовить данные в требуемом формате, либо воспользоваться готовыми скриптами для M4 и др. Реализация «как в статье» ценна тем, что включает все трюки (ансамблирование, спец. метрики и пр.) для достижения результатов статьи. Однако для быстрого прототипирования или интеграции в рабочий проект она не очень удобна. Например, чтобы спрогнозировать собственный ряд, пришлось бы писать обёртки вокруг модели и разбираться в конфигурационных .gin-файлах. Кроме того, судя по истории коммитов (всего ~5 коммитов) код практически не развивался после публикации [57] – то есть нет официальной поддержки N-BEATSx или N-HITS, мультирядности, и т.п. Тем не менее, эта реализация – *эталон точности*: она позволяет проверить, совпадают ли результаты с заявленными авторами. Лицензия репозитория – **Apache 2.0** (судя по шаблону ServiceNow Research) – что разрешает свободно модифицировать и использовать код.

Замечание по производительности. Официальный код рассчитан на обучение на GPU и может запускать множество моделей параллельно (например, на нескольких GPU с помощью команд в makefile) [58] [59]. Он также содержит примеры, как уменьшить ансамбль (поставив repeats=1 в конфиге) для быстрого прогона с минимальной потерей качества [60]. Отдельно приведены «облегчённые» результаты – авторы указали, что даже ансамбль в 10 раз меньшего размера показывает близкие показатели к полному [61].

GluonTS (MXNet, Amazon)

GluonTS – библиотека от Amazon для вероятностного прогнозирования – включает реализацию N-BEATS с 2020 г. Она написана на основе MXNet Gluon и интегрирована в общую интерфейс Estimator/Predictor библиотеки. Особенности:

- **EnsembleEstimator.** В GluonTS N-BEATS представлен в виде двух классов: `NBEATSEstimator` (одиночная модель) и `NBEATSEnsembleEstimator` (ансамбль) [20] [19]. По факту, как отмечено в доке, «The actual NBEATS model is an ensemble ... implemented by NBEATSEnsembleEstimator» [62]. То есть упор сделан на воспроизведение ансамблевого подхода авторов. В `NBEATSEnsembleEstimator` можно задать список `meta_context_length` (набор коэффициентов для длины входа) и `meta_loss_function` (список метрик – “MAPE”, “MASE”, “sMAPE”) [55] [15]. А также `meta_bagging_size` – число случайных инициализаций для каждой комбинации (по умолчанию 10) [63]. Библиотека автоматически создаёт нужное количество внутренних `NBEATSEstimator`-ов и обучает их. При прогнозировании используется `NBEATSEnsemblePredictor`, который агрегирует выходы отдельных моделей (с опцией брать медиану, среднее или выдавать все отдельно) [20] [21]. По умолчанию стоит агрегирование медианой.

- **Настройки модели.** GluonTS предоставляет достаточно гибкости в задавании гиперпараметров N-BEATS: параметры `num_stacks`, `num_blocks`, `num_layers`, ширины слоёв (`widths`), типы стеков (`stack_types`), использование шаринга весов (`sharing`) и пр. доступны в конструкторе [64] [65] [66]. При этом есть **рекомендуемые default-ы** “как в статье”: например, для generic режима – 30 стеков по 1 блоку ширины 512; для interpretable – 2 стека (trend+season) по 3 блока, ширины 256 и 2048 соответственно, с

шерингом весов между блоками тренда/сезонности [67] [65]. Метапараметры базисов: `expansion_coefficient_lengths` по умолчанию 32 для generic (т.е. длина θ-параметров 32), а для interpretable – степень полинома 3 для тренда, для сезонности этот параметр не используется [68]. Флаг `stack_types` = ["G"] или ["T","S"] выбирает generic vs interpretable соответственно [69]. Таким образом, GluonTS следует оригинальной спецификации модели очень точно.

- **Вероятностность.** Как уже упоминалось, N-BEATS в GluonTS – **исключение** среди моделей, т.к. он не основан на явном вероятностном предположении и не тренируется по likelihood. В доке прямо сказано: *"Unlike other models in GluonTS this network does not use a distribution"* [22]. Вместо этого оптимизация идёт по определённым loss-функциям, задаваемым как meta_loss_function (например, "MAPE"). Это означает, что `predict()` выдаёт детерминированные прогнозы (точечные или несколько сценариев при `aggregation_method='none'`), а не объекты распределений. Чтобы получить интервал, можно например запустить ансамбль и взять квантили по множеству прогнозов.

- **Ограничения.** Реализация GluonTS поддерживает *только univariatные ряды* (по состоянию на версии < 0.10, т.к. GluonTS на MXNet в целом ограничен univariate output для большинства моделей). Также из-за природы N-BEATS нет поддержки covariates – то есть N-BEATS нельзя скормить регрессоры в GluonTS (в то время как DeepAR или TFT в GluonTS умеют с covariates). Таким образом, GluonTS-версия покрывает именно оригинальный use-case модели. Зато она предоставляет удобный интерфейс: достаточно указать `NBEATSEnsembleEstimator(freq="D", prediction_length=H, meta_loss_function=["MAPE","MASE","sMAPE"], meta_context_length=[H*2,...,H*6])`, и библиотека сама обучит ансамбль, после чего можно вызывать `.predict()` и получать прогнозы в формате GluonTS. Для многих пользователей, уже использующих экосистему GluonTS, это значительно упрощает эксперименты с N-BEATS.

Darts (Python, PyTorch Lightning)

Darts – универсальная библиотека для работы с временными рядами (от компании Unit8) – включает реализацию N-BEATS и некоторых производных. Класс `NBEATSModel` предоставляет удобное сквозное API: создаётся модель, указываются параметры, затем вызываются `fit(series)` и `predict(n)` для получения результата. Особенности реализации в Darts:

- **Совместимость с covariates и многомерными рядами.** В документации отмечено, что помимо стандартной универиатной версии из статьи, Darts-реализация *поддерживает мультивариатные ряды* и дополнительные регрессоры [70]. Технически это сделано путём *flattening* – если входной ряд имеет D измерений, модель воспринимает его как один ряд длины $T\cdot D$, прогоняет через N-BEATS (который внутренне все равно одномерный), а затем выходной прогноз длины $H\cdot D$ реорганизуется обратно в D-мерный ряд [70]. Аналогично, допускаются **past covariates** (прошепрогнозные регрессоры): они просто конкатенируются с целевым рядом по признаку, и тоже сглаживаются в одну длинную последовательность на вход [70] [71]. Такой трюк позволяет использовать N-BEATS для мультивариатных задач, хотя это не оригинальная черта модели. Имейте в виду: flattening может не оптимально учитывать взаимосвязи между компонентами ряда, но в ряде случаев работает. Darts также поддерживает *future covariates* для некоторых моделей, но для N-BEATS судя по документации – **только past covariates** (то есть ковариаты, известные до момента прогноза, но не те, что известны на будущем горизонте) [72]. Ограничение

future covariates связано с тем, что оригинальный N-BEATS не предусматривал работу с регрессорами, и Darts не стал добавлять отдельный encoder для них. Таким образом, N-BEATS в Darts – скорее *глобальная модель* на мультивариатных рядах, чем полный аналог N-BEATSx.

- **Probabilistic forecasts.** Darts позволяет получать вероятностные прогнозы от NBEATSModel, если указать параметр `likelihood` при создании модели (например, `GaussianLikelihood()` для нормального распределения, или квантильную потерю QuantileLoss). В таком случае модель обучается не MAPE/MAE, а например по отриц. логвероятности, и на выходе может выдавать параметры распределения. Это обёртка, предоставляемая родительским классом `TorchForecastingModel` – он накладывает соответствующую функцию потерь и обеспечивает методы для выборки прогнозов. Также Darts поддерживает **Monte Carlo dropout**: если при предсказании указать `mc_dropout=True`, то even с обученной детерминированной моделью можно получить несколько разных выборок прогнозов (darts усреднит их для вероятностной оценки). Таким образом, Darts-реализация более удобна для ситуаций, когда нужны доверительные интервалы, нежели оригинальный N-BEATS.

- **Гиперпараметры.** Darts предоставляет параметр `generic_architecture=True/False`, который переключает между generic и interpretable режимом [73]. Если `False`, то всегда используется 2 стека (trend и seasonality) [74]. Степень полинома тренда задаётся `trend_polynomial_degree`, число гармоник – неявно через `expansion_coefficient_dim` (по-умолчанию 5). Можно задавать `num_stacks`, `num_blocks`, `num_layers` и т.д. – но важно: `num_stacks` имеет смысл только для generic (для interpretable игнорируется и принудительно =2) [74]. Darts выбрал значения по умолчанию несколько отличные от GluonTS: например, `num_stacks=30`, `num_blocks=1`, `num_layers=4`, ширина слоёв 256, `expansion_coefficient_dim=5`, `trend_polynomial_degree=2`. Это скорее облегчённая версия (меньше параметров), рассчитанная на более быстрый тренинг с приемлемой точностью.

- **Производительность и удобство.** Darts реализован поверх PyTorch Lightning, что упрощает тренировки на GPU, логирование и прочее. Использование выглядит так:

```
model = NBEATSModel(input_chunk_length=32, output_chunk_length=8,
generic_architecture=True)
model.fit(series_train)
forecast = model.predict(n=8)
```

  – библиотека сама позаботится о формировании обучающих сэмплов (скользящим окном) и пр. Darts также умеет обучать одну модель на **нескольких сериях** (global model): достаточно передать список TimeSeries в `fit()`. В таком случае N-BEATS научится совместно на всех (внутри Lightning batch будет содержать куски из разных серий). Для N-BEATS это нормально, особенно если ряды масштабированы. Darts, правда, не позволяет задать статические фичи на серии напрямую, так что групповая информация может быть неучтена – в отличие от N-BEATSx, где static_exog явно влияли на разделяемые параметры.

В целом, Darts предоставляет наиболее **пользовательски-дружественный** интерфейс к N-BEATS: можно быстро попробовать модель на своих данных, комбинировать с другими моделями,

делать ансамбли (Darts имеет класс EnsembleModels для усреднения прогноза разных методов). Лицензия Darts – **Apache-2.0** [75].

Nixtla NeuralForecast (Python, PyTorch)

Nixtla – компания, развивающая open-source инструменты для прогнозирования. В их библиотеке **NeuralForecast** (ранее называлась ETS, потом MLForecast, сейчас объединено) представлены *эталонные реализации* многих нейросетевых моделей, включая N-BEATS и расширения. Особенности реализации Nixtla:

- **Полная линейка моделей.** NeuralForecast включает `NBEATS`, `NBEATSx` и `NHITS` как отдельные классы [76] [77]. Причём указывается, что NBEATSx и NHITS – *официальные реализации, предоставленные авторами* (N-BEATSx – IJF 2023, NHITS – AAAI 2023) [37]. Разработчики Nixtla (Cristian Challu и др.) сами соавторы этих моделей, поэтому их код можно считать референсным. В отличие от Darts/GluonTS, здесь все последние улучшения наличествуют "из коробки". Например, `NBEATSx` класс имеет параметры `futr_exog_list`, `hist_exog_list`, `stat_exog_list` для передачи имен колонок экзогенных регрессоров (исторических, будущих и статических) [78]. `NHITS` класс – с параметрами multi-rate: списки `n_pool_kernel_size`, `n_freq_downsample` и пр. для настройки pooling по стекам [79] [40].

- **Единый интерфейс и доп. возможности.** NeuralForecast организован по принципу sklearn/pytorch-forecasting: создаётся объект `NeuralForecast(models=[...], freq=freq)`, затем вызывается `fit(df)` и `predict()`, которые работают сразу по всему списку моделей и по всем рядам датасета (в формате long DataFrame) [80] [81]. То есть, библиотека ориентирована на обучение *глобальных моделей* на множестве рядов, с поддержкой идентификаторов серий. Это удобно для индустриальных задач, где нужно прогнозировать, например, 1000 товаров – можно обучить один N-BEATS на все сразу, указав `df` с колонкой `unique_id` и, например, добавить `stat_exog` (категории товаров). Nixtla автоматически параллелит computation, использует PyTorch Lightning DDP для распределённого обучения. Также встроены функции масштабирования, кросс-валидации, backtest, и даже автоматический подбор гиперпараметров (Optuna/RayTune) [82].

- **Продвинутая функциональность.** Библиотека Nixtla изначально создавалась, чтобы восполнить пробел между исследованиями и production: *"available implementations are hard to use and often fail to outperform stats methods"* [83]. Поэтому они акцентируются на *эффективности* (c++, numba в критических местах, батчинг), *поддержке вероятностного прогноза* (любой модели можно задать параметр `loss=QuantileLoss(qs=[...])` или `loss=MQMELoss()` и она будет обучена по квантилям или CRPS – для N-BEATS это тоже применимо). В NeuralForecast есть адаптеры для **интерпретации**: например, для N-BEATS интерпретируемого можно запросить разложение прогнозов по компонентам (trend/seasonal). Также Nixtla поддерживает *предсказание с малой историей посредством transfer learning* – например, используя предварительно обученные N-HiTS (т.н. модели TimeGPT/TinyTimeMixer), но это отдельная тема.

- **Отличия реализации.** N-BEATS в Nixtla имеет по умолчанию конфигурацию: `stack_types=['identity','trend','seasonality']` с соответствующими параметрами [84]. То есть они используют **гибридную конфигурацию**: три стека – один generic (identity), один тренд, один сезон. Вероятно, это сделано для сочетания

преимуществ (первый стек берет на себя произвольные эффекты, два других – гарантируют уловимые тренды/сезон). Подобный вариант не описан в исходной статье, но возможен, и Nixtla считает его разумным default. Параметры по умолчанию: по 1 блоку в каждом стеке, все блоки по 2×512 нейронов, полином степени 2, 2 гармоники [85]. Эти значения можно менять.

Суммируя, **Nixtla NeuralForecast** – наиболее полнофункциональная и актуальная реализация: она включает N-BEATS, N-BEATSx, N-HITS, с поддержкой экзогенов, вероятностных потерь, GPU-обучения и т.д., и хорошо документирована (с туториалами на nixtlaverse.nixtla.io). Лицензия – Apache-2.0 [86].

Прочие реализации и библиотеки

Помимо вышеупомянутых, существуют и другие open-source коды:

- Независимые переводы на Keras/TensorFlow – например, проект philipperemy/n-beats, который появился вскоре после публикации. Он предоставляет реализацию на Keras и PyTorch с MIT-лицензией [87]. Эта версия была полезна в 2019–2020, когда официального кода не было, но сейчас уступает по возможностям библиотечным.

- PyTorch Forecasting – популярная библиотека от Jan Gasthaus (AWS) – как ни странно, **не содержит N-BEATS** (на момент 2023 г.). Она фокусируется на рекуррентных (LSTM), TFT и CNN (TCN) моделях, но N-BEATS почему-то не был добавлен.

- Sktime (Python) – библиотека для статистического обучения временны́м рядам – имеет интерфейсы для глубоких моделей. Насколько известно, прямой реализации N-BEATS там нет, но есть обёртка `NeuralForecaster` для Nixtla, которая позволяет использовать Nixtla-модели как sktime-эстиматоры.

- TorchTS, PyTorchTS – некоторые небольшие пакеты, но они менее популярны.

Сравнение API и возможностей. Ниже приведена таблица, сравнивающая ключевые реализации:

Реализация	Фреймворк	Лицензия	Особенности и функции	Ограничения
Офиц. (ServiceNow) [49] [50]	PyTorch (скрипты)	Apache 2.0	Точное воспроизведение статьи; есть ансамбли, конфиги для M3/M4	Нет простого API, только скриптовый запуск; без поддержки экзогенов; не обновляется

Реализация	Фреймворк	Лицензия	Особенности и функции	Ограничения
GluonTS [55] [56]	MXNet/ Gluon	Apache 2.0	Эстиматор + ансамбль; автосемплинг данных; конфиги generic/interp [67] [66]	Только универиатный прогноз; детерминированные выходы (по точечным метрикам) [22]
Darts [70]	PyTorch (Lightning)	Apache 2.0	Лёгкий high-level API; поддержка мультивариатных рядов и ковариат flatten-методом [70]; вероятностный прогноз (likelihood/ MC-dropout)	Flatten может быть неэффективен при многих сериях; нет явного разделения тренд/сезон; future covariates не поддерживается
Nixtla NeuralForecast [37] [88]	PyTorch (Lightning)	Apache 2.0	Поддержка всей семейства (N-BEATS, N-BEATSx, N-HITS); единый интерфейс `.fit/.predict` для многих серий; экзогенные и статические переменные [88]; интерпретируемость компонентов; выбор различн. функций потерь (MAE, quantiles, MAPE и т.д.)	Менее известна широкой аудитории (новая библиотека); для одиночного ряда требует форматировать DataFrame с идентификаторами (не pure numpy)

Как видно, **выбор реализации** зависит от задач: для быстрого пилота на одном ряду удобен Darts; для масштабной задачи с большим числом серий и экзогенами – Nixtla; для воспроизведения академических результатов – официальный код или GluonTS. Отметим, что между разными реализациями могут быть небольшие отличия “по умолчанию” (разный набор слоёв, масштабирование данных и т.п.), поэтому результаты могут различаться. Например, Darts по умолчанию масштабирует серии к диапазону [0,1] перед обучением, GluonTS – использует дифференцируемое масштабирование (ScaleNormalizer) внутри, Nixtla – позволяет явно указать тип скейлера (`scaler_type` параметр, default identity) и т.д. Все реализации согласуются в одном: **модель N-BEATS достаточно ресурсоёмкая**, особенно в ансамбле – на CPU обучение может быть очень долгим, поэтому на практике почти всегда используется GPU.

4. Сравнение N-BEATS и современных моделей временных рядов

N-BEATS и её потомки (N-BEATSx, N-HITS) представляют собой семейство чисто нейросетевых *autoregressive* моделей. На момент выхода (2020) N-BEATS стала значимым прорывом, однако с тех

пор появилось множество других подходов. Рассмотрим сравнение с некоторыми state-of-the-art методами:

- **DeepAR (RNN).** DeepAR (2017, Amazon) – это авторегрессионная модель на основе рекуррентной сети LSTM, которая генерирует вероятностный прогноз через семплирование. По сравнению с N-BEATS, DeepAR естественно поддерживает ковариаты и категориальные особенности (через feeding в LSTM на каждом шаге) и выдаёт полный вероятностный прогноз. DeepAR стал стандартом для *глобального прогнозирования* многих рядов сразу. N-BEATS (особенно расширенный N-BEATSx) способен выполнять ту же задачу глобального обучения, причём обычно достигает *бóльшей точности*: в M4 N-BEATS превзошёл DeepAR примерно на 15% по sMAPE [89]. Причины – MLP-архитектура с большим числом параметров оказывается более гибкой, а отсутствие рекуррентности устраняет проблему накопления ошибки при многошаговом прогнозе (DeepAR вынужден предсказывать точку за точкой, ошибаясь на каждом шаге). Однако DeepAR лучше в *генерации распределений*: N-BEATS необходимо прибегать к энsembling или квантильным потерям для получения интервалов, в то время как DeepAR напрямую моделирует вероятностное распределение (обычно через параметрическую смесь). По **интерпретируемости** DeepAR – "чёрный ящик" (LSTM весы сложно трактовать), тогда как N-BEATS может явно дать трендовую/сезонную составляющую (в интерпретируемом режиме) [9]. Также N-BEATS обычно быстрее обучается на коротких сериях, т.к. нет внутренней развёртки по времени – вся история прокидывается через сеть параллельно. Но на очень длинных исторических окнах (скажем, 1000+ точек) LSTM может быть выгоднее, т.к. MLP рост параметров линейный в длину входа, а LSTM – только квадратичный во внутреннем размере.

- **TCN и прочие CNN-модели.** Temporal Convolutional Network (Bai et al., 2018) – семейство свёрточных архитектур, использующих раздвоенные (dilated) сверки для улавливания долговременных зависимостей. В задачах прогнозирования TCN применялся (например, модель LSTNet 2018 комбинировала CNN+RNN). В принципе, TCN аналогичен N-BEATS по идее: он тоже не рекуррентен, захватывает длинный контекст параллельно. Разница – TCN сохраняет локальность признаков (свертки окнами), в то время как N-BEATS использует глобально соединённый MLP, что позволяет любому входному времени влиять на любой выходной. На практике N-BEATS обычно превосходит простые TCN по точности (так отмечается, например, в работах по энергопрогнозированию [90] [91]). Однако TCN нашёл применение внутри N-BEATSx как экзогенный энкодер – это показатель, что комбинация MLP + CNN может быть плодотворной. По скорости: TCN может быть более экономичен, ведь свёртки имеют меньше параметров чем полносвязный слой на тот же вход (особенно если вход большой). Но с другой стороны, чтобы достичь такой же гибкости, нужно много фильтров и слоёв. N-BEATS vs TCN – это своего рода борьба "глобального" (MLP) и "локального" (CNN) подходов. В современных решениях иногда объединяют их: например, модели типа *N-Beats-RNN* или *AR-DE N-BEATS* добавляют CNN/RNN-элементы к основе N-BEATS [92]. N-BEATS явно выигрывает у TCN в объяснимости – последние не предоставляют явной декомпозиции.

- **Transformer-подходы (TFT, Informer, Autoformer, PatchTST и др.).** С 2019 г. в прогнозировании стали активно использовать механизмы self-attention, заимствованные из NLP. **Temporal Fusion Transformer (TFT)** (Lim et al., 2019) – гибрид LSTM + Transformer: он обрабатывает исторические данные LSTM'ами, а важность ковариат оценивает через attention-механизмы. **Informer** (Zhou et al., 2021) – облегчённый Transformer для очень длинных последовательностей, вводящий спарс-просеивание внимания (ProbSparse) [93]. **Autoformer, FEDformer** (2021) – улучшения attention с фокусом на автокорреляции и

разложении частот. **PatchTST** (Nie et al., 2022) – недавно предложенный метод: разбивает ряды на «патчи» и применяет трансформер к патчам, избегая обрабатывать каждую точку напрямую. **TinyTimeMixer** (Yu et al., 2023, IBM) – не совсем трансформер, а MLP-Mixer, но тоже с идеей предварительного обучения на большом наборе рядов, позиционируется как "foundation model" для рядов [94].

Сравнение с трансформерами следует разбить на аспекты:

Точность. Трансформеры изначально не сразу превосходили N-BEATS. Например, TFT давала хорошее качество на задачах с множеством ковариат (продажи, экономика), но на чисто временных сериях (M4) не показала заметного выигрыша. Informer заявлял превосходство на длинных горизонтах (например, 24 часа нагрузки), но Challu et al. отметили, что N-HiTS превзошёл Informer на 25% [45]. Последние модели (PatchTST) приблизили трансформеры к SOTA, показав лучшее качество на ряде бенчмарков (ETTh1, ETTm1, Weather) по сравнению с N-HiTS. Однако разрыв невелик – PatchTST и N-HiTS в пределах нескольких процентов, периодически обгоняя друг друга на разных наборах. В целом, *трансформеры сильны на высокоразмерных многомерных сериях*, где их способность дифференцировать связи между множеством входных признаков становится критична (например, одновременное прогнозирование 100 варьирующихся фич). Для чисто одномерных рядов преимуществ может не быть – простая MLP зачастую обыгрывает излишне гибкий attention, страдающий от переобучения на шум.

Скорость и ресурсы. Здесь N-BEATS/N-HiTS вне конкуренции – трансформеры очень тяжеловесны. Как отмечалось, N-HiTS даёт ~50x ускорение против Informer [46], и даже простая N-BEATS без оптимизаций часто обучается быстрее, чем трансформер с аналогичным количеством параметров (за счёт меньшей последовательности операций – нет квадратичного внимания). Трансформеры требовательны к памяти (квадратичная по длине ввода) – поэтому они изобретают эвристики (Informer, etc.). N-BEATS обучаясь на фиксированном окне, может обрабатывать окна сколь угодно длинного ряда последовательно, не требуя памяти > окно. Таким образом, в сценариях с ограниченными ресурсами (например, edge-устройства) N-BEATS и его потомки предпочтительнее: TinyTimeMixers вообще показывают, что **маленькие MLP-модели (<1 млн параметров) могут быть предобучены и работать универсально** [95]. IBM TinyTimeMixer, например, заявлен как способ превзойти большие модели буквально тысячей параметров.

Экзогенные факторы. Трансформерные модели превосходны в учёте большого числа разнотипных регрессоров: TFT может одновременно оперировать статическими категориальными признаками через embedding, known-future covariates через отдельный self-attention блок, and past observed covariates – всё это влиять на выход. Если задача, например, прогноз розничных продаж с промо-акциями, погодой, экономикой – TFT предоставит более прямой способ учесть эти данные, чем N-BEATSx (где нужно вручную строить соответствующие экзогенные входы). С другой стороны, TFT сильно сложнее и требователен к данным: для успешного обучения нужно большое количество наблюдений, иначе он переобучится. N-BEATSx проще и может быть устойчивее на умеренных объемах данных (интерпретируемый режим служит регуляризатором).

Интерпретируемость. Здесь интересно: **TFT** позиционировался как “interpretable” Transformer, т.к. выдаёт важности атрибутов (веса внимания к каждому ковариату), важности временных лагов, и имеет механизм Variable Selection Network. Однако интерпретация TFT – скорее feature importance, в то время как **N-BEATS** даёт структурную интерпретацию (сколько в прогнозе тренда, сколько сезонности). В бизнес-приложениях часто понятнее увидеть прогноз в виде: “прогноз = базовый тренд + сезонные колебания”, чем “важность фич: промо=34%, праздник=10%...”. Таким

образом, *с точки зрения объяснения поведения ряда во времени N-BEATS интерпретируемый может быть предпочтительнее*. Более того, N-BEATS можно применить постфактум к серии: обучить в interpretable-режиме и получить графики тренда и сезон, что фактически делает её инструментом для анализа временного ряда (в духе классического STL-декомпозера, но обученного нейросетью).

Обучаемость. Нейросети иногда страдают от нестабильности обучения (vanishing gradients, sensitivity to lr). N-BEATS с residual-блоками оказался очень устойчивым – авторы сообщали, что он *fast to train* даже без специального тюнинга [23]. Трансформеры же пресловуты требовательностью к настройке (warm-up schedule, layer normalization, инициализация). В конкурсах (например, M5) классические модели/градиентный бустинг побеждали во многом из-за простоты – их легче было довести до рабочего состояния, чем громоздкий трансформер. N-BEATS в этом отношении ближе к "традиционным" – его можно запустить с дефолтными параметрами и он выдаст разумный прогноз (особенно interpretable версия с априорными базисами).

- **Другие новые архитектуры.** Помимо уже упомянутых, есть гибридные подходы: например, модели, комбинирующие статистические методы с нейросетями (ESRNN – эксп. сглаживание + RNN; ProphetNet; ThetaRNN и др.). N-BEATS изначально доказал, что чистая нейросеть может превзойти такие комбинации – выигрыш +3% над ESRNN на M4 [96]. Но гибриды всё ещё популярны, особенно когда нужен интерпретируемый компонент. Другой тренд – **Neural ODE / дифференциальные модели** (NeuralProphet, 2021) – они стремятся интегрировать априорные тренд-сезон-фичи в обучаемую модель. По сути, N-BEATS interpretable – это частный случай такой интеграции (гармоники+полиномы). Можно отметить также **GBRT и CatBoost с фичами лага** (классические модели) – на некоторых Kaggle это побеждает DL, но требуя дорогостоящего ручного генератора признаков. N-BEATS автоматизирует это, поэтому в равных условиях обычно точнее.

Преимущества N-BEATS и производных:

- *Универсальность.* Могут применяться ко множеству доменов без изменения (достаточно подать сырые ряды) [23]. В то время как специализированные модели (ARIMA, ETS) требуют стационарности или других предположений.

- *Высокая точность на разнородных наборах.* Проверена на конкурсах M3/M4 – N-BEATS работает хорошо и на годовых макроэкономических рядах, и на месячных финансовых, и на минутных потоках – единая модель закрывает всё [1].

- *Интерпретируемость.* Возможность явного разложения прогноза по компонентам – редкая черта для нейросетей. Это ценится в прикладных задачах (принятие решений, доверие бизнес-пользователей).

- *Отсутствие сезонных гиперпараметров.* Модель не требует указывать, есть ли годовая сезонность, недельная – она способна сама их выявить (в interpretable режиме через обучение коэффициентов при гармониках). Это избавляет от ручной инженерии.

- *Хорошая поддержка глобального обучения.* N-BEATS легко обучить на тысячах серий сразу – экономия от этого колоссальна (обучение одной модели вместо тысячи отдельных). Модель выясняет общие паттерны и переносит знания между рядами, что улучшает точность для коротких историй.

- *Вычислительная эффективность.* По сравнению со сложными архитетурами (CNN+Transformer), MLP-блоки очень быстры на GPU, а N-HITS делает их ещё быстрее за счёт сокращения входов через pooling [41] [97]. Для сценария edge-deployment можно сильно урезать сеть, пожертвовав минимумом точности – вплоть до TinyTimeMixer (упрощённая N-BEATS) с <1M параметров, способной всё ещё показывать SOTA уровень [94].

- *Простота реализации.* Полносвязная сеть – одна из самых понятных для разработчика, много поддерживающих библиотек. Порог входа ниже, чем для тех же трансформеров.

Недостатки и ограничения:

- *Одновременное прогнозирование многих переменных.* В оригинале N-BEATS – униварианая модель. Расширения (Darts flatten, Nixtla multivariate MLP) есть, но всё же, если нужно предсказать сразу вектор из 10 выходных переменных (скажем спрос на 10 товаров взаимосвязанных) – трансформер или многомерный VARMLP могут справиться лучше, учитывая межсерийные корреляции. N-BEATS в такой ситуации придётся либо строить отдельную модель на каждый, либо пытаться вместе (flatten), что не всегда оптимально.

- *Экзогенные ковариаты до N-BEATSx.* Без N-BEATSx, модель не умеет учитывать известные наперёд события – это большой минус (который, правда, устраняется использованием N-BEATSx). К 2025 г. уже трудно представить модель без поддержки экзогенов, так что оригинальный N-BEATS устарел в этом аспекте.

- *Длинный прогноз без N-HITS.* Аналогично, оригинальный N-BEATS может хуже справляться с $H\gg T$ ситуациями. Например, прогноз на 3 года вперёд при истории 1 год – трансформеры с их механизмами фьюзинга в принципе могут экстраполировать паттерны, а MLP может выдать неустойчивый результат (если, конечно, не добавить априорных компонентов). N-HITS сильно улучшил это, но если его не применять, N-BEATS может проигрывать.

- *Ансамблирование как необходимость.* Для лучшей точности N-BEATS полагается на ансамбли из десятков моделей, что увеличивает вычислительную стоимость линейно. Другие методы (та же Prophet или модификации Transformers) часто могут работать удовлетворительно без ансамблей. Правда, позже появились приёмы получать вероятность и устойчивость иначе – но классический N-BEATS “из коробки” подразумевает 180 сетей, что малопригодно вне соревнований.

- *Обучение по точечным метрикам.* Оптимизация по MAPE/sMAPE – нестандартная для DL (не гладкая функция). Это могло осложнять сходимость, требовать специальных трюков. Многие современные DL-модели обучаются по MSE (а потом результаты оценивают по MAPE). N-BEATS же гнался именно за улучшением sMAPE, что хорошо для соревнований, но может быть избыточным (переоптимизация под одну метрику может ухудшать другую). Поэтому в практиках часто переобучают N-BEATS на MSE, если нужна стабильность.

- *Отсутствие встроенной иерархичности/ограничений.* Некоторые задачи требуют, например, чтобы прогноз соблюдал агрегатные ограничения (иерархический прогноз: сумма низов = верх). N-BEATS не учитывает это априори. Хотя существуют работы, пытающиеся привнести иерархичность (например, модель HINT – Hierarchical Interpolation, 2022 – кстати, Nixtla реализовала HINT как пост-обработку для любого метода).

Заключение по сравнению: N-BEATS занял прочное место в арсенале прогнозиста. На 2025 г. он остаётся одним из *baseline*-методов, с которыми сравнивают новые разработки. Например, PatchTST, Informer в своих статьях всегда приводят сравнение с N-BEATS или N-HITS и часто признают, что MLP-модель выступает на уровне или лучше при гораздо меньшей сложности [98] [45]. Можно сказать, что N-BEATS реабилитировал простые нейросетевые архитектуры в домене, где долго царили либо сложные RNN, либо громоздкие трансформеры. В различных сценариях N-BEATS может уступать: если задача требует глубокого понимания внутренних связей (например, предсказать одновременное поведение десятков связанных временных рядов) – тогда многомерные модели или факторные модели предпочтительнее. Если главная цель – строго вероятностные сценарии (например, value-at-risk в финтехе) – вероятностные модели (ARIMA-GARCH, DeepAR, TFT с quantile loss) могут дать более обоснованные распределения. Однако по метрикам точности прогноза (MAE, sMAPE) N-BEATS и его потомки зачастую либо на первом месте, либо в топ-3 для большинства публичных наборов данных на сегодняшний день.

5. Признание, лицензии и использование в сообществе

Публикации и цитируемость. Статья *"N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting"* была представлена на ICLR 2020 – одной из ведущих конференций по машинному обучению [99]. С момента публикации модель получила большое внимание исследователей и практиков. По данным Google Scholar, работа Oreshkin et al. (2020) имеет несколько сотен цитирований (более 600 упоминаний по состоянию на 2023) – это говорит о её влиянии в области [100]. N-BEATS часто упоминается как новый бенчмарк для сравнения, его включают в обзоры методов прогноза. В 2020–2021 появилось несколько производных работ: например, про использование N-BEATS для среднесрочного прогноза нагрузки на электроэнергии [101], для финансовых временных рядов, для COVID-19 кейсов и др. Многие из этих исследований подтверждают высокую точность модели и адаптируют её к своим доменам (иногда комбинируя с эволюционными алгоритмами, смешанными эффектами и т.п.).

Расширение **N-BEATSx** было первоначально опубликовано как препринт arXiv в 2021, а затем в 2023 принято в **International Journal of Forecasting** – престижный журнал по прогнозированию [102]. Оно ориентировано на энергетический домен (прогноз цен на рынке электричества) и привлекло внимание энергетиков; сейчас цитируется в работах по энергорынкам. **N-HiTS** вышел как препринт в 2022, получил положительные отзывы и был принят на **AAAI-2023** (одна из топ-конференций ИИ) [98]. За N-HiTS последовал ряд статей, сравнивающих её с трансформерами (например, конкурсные статьи, где N-HiTS и N-BEATS анализируются для финансовых рядов [103]). Можно сказать, Nixtla-команда успешно продвигает эту линейку моделей в академическом сообществе: N-HiTS получила награду **Best Paper Award на AAAI 2023 (трек времян. ряды)**, если верить Nixtla (в документации у них так упомянуто) [45].

Лицензии и открытость. Все основные реализации N-BEATS доступны под открытыми лицензиями. Официальный код – Apache 2.0 (ServiceNow открыла многие проекты под Apache) [104]. Реализации GluonTS, Darts, Nixtla – тоже Apache-2.0 (или MIT), что позволяет без ограничений использовать их в коммерческих приложениях. Важно заметить, что хотя код открыт, **патентных ограничений** в области ML обычно нет – т.е. свободно можно применять N-BEATS в продуктах (исключая случаи, если бы был какой-то патент, но такого не известно).

Сообщество и развитие. После 2020 г. авторы оригинальной работы (Boris Oreshkin и коллеги) продолжили заниматься временем рядами: Oreshkin участвовал в разработке N-HiTS [105], а в 2022 перешёл в Amazon (Principal Scientist) – вероятно, его опыт внедряется в продукты AWS Forecast. Nicolas Chapados – ветеран индустрии, сооснователь ElementAI, сейчас в ServiceNow – возможно,

N-BEATS лег в основу каких-то сервисов прогнозирования ServiceNow. Yoshua Bengio, соавтор, известен по работам в deep learning, для него N-BEATS был интересным кейсом применения DL к новой области.

Независимое сообщество (Nixtla, Unit8, etc.) активно поддерживает модели: выходят новые версии библиотек, исправляются баги. В Nixtla `neuralforecast` репозиторий на GitHub имеет регулярные коммиты, а также Slack-канал, где можно задать вопросы. Darts тоже развивается, добавляя модели – они включили N-HiTS в вер. 0.22.

Соревнования. N-BEATS появился после завершения **M4 (2018)**, но авторы продемонстрировали пост-фактум, что их подход превзошёл победителя M4 [1]. В **M5 (2020)** – конкурсе по розничным продажам – top-решения представляли ансамбли статистических моделей и LightGBM, однако некоторые команды экспериментировали с N-BEATS. Например, в обсуждениях Kaggle отмечали, что *"N-BEATS could beat M4 winner's score"*, но для M5 требовалась иерархическая согласованность и множество промо-фич, где из коробки N-BEATS уступал [106]. Тем не менее, идеи N-BEATS внедрялись: одна из топовых команд M5 использовала гибрид из бустингов и нейросетей, где нейросетевой компонент напоминал N-BEATS (глубокий residual MLP). В последующих конкурсах (например, M6 – прогноз финансовых индикаторов) N-BEATS применялся как бенчмарк, но специфика M6 (прогноз доходностей акций) меньше подходит для чисто временных моделей. Зато в отраслевых соревнованиях (GEFCom, ERCOT Load Forecast) версии N-BEATSx показывали отличные результаты, учитывая погодные экзогены.

Применения на практике. На 2025 год N-BEATS и родственники внедряются в разных компаниях: Amazon в своем сервисе *GluonTS/SageMaker Forecast* предлагает N-BEATS как одну из моделей; Nixtla предоставляет SaaS API *TimeGPT* (прогноз "как сервис"), где под капотом несколько моделей, включая N-BEATSx/N-HiTS. В банкинге и ритейле N-BEATS ценят за точность, а интерпретируемую версию – за доверие: например, в supply chain прогнозах модель, показывающая "вот тренд растёт, вот сезонный пик в праздники" вызывает больше доверия у планировщиков, чем абстрактные числа от LSTM.

Вывод: N-BEATS прошла путь от научной идеи к широко используемому инструменту за короткое время. Её архитектурные принципы – глубокие residual-блоки MLP и использование базисных функций – вдохновили множество работ. За прошедшие 5 лет к модели добавились важные способности (работа с внешними факторами, многомасштабность), сохранив при этом изначальную простоту. Сегодня N-BEATS и её модификации входят в число **state-of-the-art моделей прогнозирования** наряду с трансформерами, и выбор между ними диктуется скорее практическими соображениями (доступные данные, требование интерпретации, вычислительный бюджет), чем уровнем точности – который у всех этих моделей очень высок. Можно ожидать, что дальнейшее развитие пойдёт по пути синтеза идей: возможно, появятся модели, сочетающие интерпретируемость N-BEATS с мощью трансформеров, или использующие pretrained N-BEATS-основу для решения zero-shot задач (как TinyTimeMixers). Но уже сейчас ясно, что вклад N-BEATS в область прогнозирования временных рядов стал исторически значимым, а сама модель – рабочей лошадкой для множества практических приложений.

Источники:

- Oreshkin et al. (2020). *"N-BEATS: Neural basis expansion analysis for interpretable time series forecasting."* ICLR 2020 [1] [3].
- B. Oreshkin, D. Carpov, N. Chapados, Y. Bengio (2019). *Apxiv preprint 1905.10437*. Код ServiceNow (Apache 2.0) [49] [50].

- GluonTS Documentation – *NBEATSEnsembleEstimator* [55] [56], конфигурация стеков [67] [66].
- Challu et al. (2023). *"NHITS: Neural Hierarchical Interpolation for Time Series Forecasting."* AAAI-23 [45] [41]. Код Nixtla (Apache 2.0).
- Olivares et al. (2021). *"NBEATSx: Neural basis expansion analysis with exogenous variables."* IJF 2023 [25] [26].
- Nixtla docs: N-BEATS [8] [2], N-BEATSx [107] [36], N-HITS [42] [45].
- Unit8 Darts docs: N-BEATSModel [70] [73] (поддержка мультивариатности и пр.), N-HiTSModel.
- Kaggle M5 discussion (2020): упоминание N-BEATS [106].
- Zhang et al. (2022). *"A Comparative Analysis of Neural Forecasting Models N-HiTS and N-BEATS in Finance."* (arXiv) [103].
- Yu et al. (2023). *"Tiny Time Mixers: Efficient Pretrained Models for TS Forecasting."* (IBM Research) [94].
- Другие ссылки на репозитории и статьи, указанные по тексту.

---

1 3 12 23 N-BEATS: Neural basis expansion analysis for interpretable time series forecasting | OpenReview
https://openreview.net/forum?id=r1ecqn4YwB

2 8 9 10 13 84 85 96 NBEATS - Nixtla
https://nixtlaverse.nixtla.io/neuralforecast/models.nbeats.html

4 5 6 7 A Detailed Explanation of the Workflow of N-BEATS Architecture | by Adria Binte Habib | Medium
https://adria708.medium.com/a-detailed-explanation-of-the-workflow-of-n-beats-architecture-9f20f5a1b3ba

11 15 16 17 18 19 20 21 22 55 56 62 63 64 65 66 67 68 69 gluonts.mx.model.n_beats package - GluonTS documentation
https://ts.gluon.ai/stable/api/gluonts/gluonts.mx.model.n_beats.html

14 36 76 77 78 107 NBEATSx - Nixtla
https://nixtlaverse.nixtla.io/neuralforecast/models.nbeatsx.html

24 25 26 27 28 29 30 31 32 33 34 35 89 102 [2104.05522] Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx
https://ar5iv.org/abs/2104.05522

37 80 81 82 83 86 88 GitHub - Nixtla/neuralforecast: Scalable and user friendly neural forecasting algorithms.
https://github.com/Nixtla/neuralforecast

38 44 46 [2201.12886] N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting
https://arxiv.org/abs/2201.12886

39 40 41 42 43 45 47 79 93 97 98 105 NHITS - Nixtla
https://nixtlaverse.nixtla.io/neuralforecast/models.nhits.html

48 49 50 51 52 53 54 57 58 59 60 61 99 GitHub - ServiceNow/N-BEATS: N-BEATS is a neural-network based model for univariate timeseries forecasting. N-BEATS is a ServiceNow Research project that was started at Element AI.
https://github.com/ServiceNow/N-BEATS

70 71 73 74 N-BEATS — darts documentation
https://unit8co.github.io/darts/generated_api/darts.models.forecasting.nbeats.html

72 Darts Forecasting Deep Learning & Global Models - Kaggle
https://www.kaggle.com/code/ferdinandberr/darts-forecasting-deep-learning-global-models

75 Time Series Made Easy in Python - darts · PyPI
https://pypi.org/project/darts/0.8.1/

87 GitHub - philipperemy/n-beats: Keras/Pytorch implementation of N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.
https://github.com/philipperemy/n-beats

90 91 N-Beats architecture for explainable forecasting of multi-dimensional poultry data - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC12021150/

92 N-BEATS-RNN: deep learning for time series forecasting - IEEE Xplore
https://ieeexplore.ieee.org/document/9356308

94 95 Tiny Time Mixers(TTMs): Powerful Zero/Few-Shot Forecasting ...
https://aihorizonforecast.substack.com/p/tiny-time-mixersttms-powerful-zerofew

100 Boris N. Oreshkin - Google Scholar
https://scholar.google.ca/citations?user=48MBCeIAAAAJ&hl=en

101 N-BEATS: Neural basis expansion analysis for interpretable time ...
https://arxiv.org/abs/1905.10437

103 A Comparative Analysis of Neural Forecasting Models N-HiTS and ...
https://arxiv.org/html/2409.00480v1

104 Repositories - ServiceNow - GitHub
https://github.com/orgs/ServiceNow/repositories

106 M5 Forecasting - Accuracy - Kaggle
https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/133551