НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет/(ННІ) _____інформаційних технологій_____

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ)	**Завідувач кафедри**
____інформаційних технологій_____ (назва факультету (ННІ))	_____комп'ютерних наук_____ (назва кафедри)
_____ (підпис) ____Ігор БОЛБОТ_____ (ім'я ПРІЗВИЩЕ)	_____ (підпис) __Белла ГОЛУБ___ (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.	"_28_" _листопада_ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему __Дослідження ефективності трансформерних архітектур (BERT, GPT) у прогнозуванні цін фінансових активів на основі новинного фону_____

Спеціальність _____121 «Інженерія програмного забезпечення»_____
(код і найменування)

Освітня програма ____Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____к.ф-м.н. доцент_____ _____ ____Віктор КИРИЧЕНКО___
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи

_____д.т.н., професор_____ _____ _____Наталія ЗАЄЦЬ_____
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ _____Владислав МАСЕНКОВ_____
(підпис) ((ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
__к.т.н., доцент__ _____ ___Белла ГОЛУБ___
(науковий ступінь, вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)
"_27_" _____жовтня_____ 2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

_____Масенкоу Владиславу Ігоровичу_____
(прізвище, ім'я, по батькові)

Спеціальність _____121 «Інженерія програмного забезпечення»_____
(код і найменування)

Освітня програма _____Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи_____Дослідження ефективності трансформерних архітектур (BERT, GPT) у прогнозуванні цін фінансових активів на основі новинного фону_____
затверджена наказом від "_27_" __жовтня__2025 р. №_2554"С"_

Термін подання завершеної роботи на кафедру _____28 листопада 2025 р._____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____

1. часові ряди цін фінансових активів (акцій, індексів, криптовалют)_____
2. корпус новин та аналітичних матеріалів, що стосуються вибраних фінансових активів_____

Перелік питань, що підлягають дослідженню:

1. Дослідження сучасних підходів та можливостей використання новинного фону у прогнозуванні фінансових активів._____
2. Аналіз архітектур LSTM та трансформерних моделей (BERT, FinBERT, GPT) і дослідження їх придатності для фінансового прогнозування._____
3. Можливості інтеграції часових рядів та новинних текстів у гібридні моделі.___
4. Вплив параметрів навчання та ринкової волатильності на точність і стабільність прогнозів._____
5. Аналіз результатів моделей та оцінка їх ефективності за метриками точності, стабільності та практичної застосовності._____

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "__27__"____жовтня____2025 р.

Керівник магістерської кваліфікаційної роботи _____Наталія ЗАЄЦЬ_____
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ ____Владислав МАСЕНКОВ_____
(підпис) (ім'я ПРІЗВИЩЕ)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI	Штучний інтелект (*Artificial Intelligence*);
BERT	Двонапрямлена модель кодування на основі трансформера (*Bidirectional Encoder Representations from Transformers*);
CNN	Згорткова нейронна мережа (*Convolutional Neural Network*);
EMA	Експоненційне ковзне середнє (*Exponential Moving Average*);
FinBERT	Фінансова модифікація моделі BERT (*Financial BERT*);
GPT	Генеративний попередньо натренований трансформер (*Generative Pre-trained Transformer*);
GPU	Графічний процесор (*Graphics Processing Unit*);
LSTM	Мережа довгої короткочасної пам’яті (*Long Short-Term Memory*);
MAE	Середня абсолютна похибка (*Mean Absolute Error*);
MAPE	Середня абсолютна відносна похибка (*Mean Absolute Percentage Error*);
ML	Машинне навчання (*Machine Learning*);
MLP	Багатошаровий перцептрон (*Multi-Layer Perceptron*);
RMSE	Середньоквадратична похибка (*Root Mean Square Error*);
TFT	Трансформер часових рядів (*Temporal Fusion Transformer*);

ВСТУП

У сучасних фінансових ринках, де мільйони угод здійснюються щосекунди, ключовими чинниками успішного прогнозування є швидкість обробки даних, здатність виявляти приховані закономірності та врахування контексту, що формується інформаційним середовищем.

Традиційні статистичні моделі, такі як ARIMA, VAR чи GARCH, виявляються недостатньо ефективними для опису нелінійних взаємозв'язків та адаптації до динамічних умов ринку.

Водночас розвиток технологій глибинного навчання (Deep Learning) відкрив нові можливості аналізу часових рядів та текстових даних у єдиному інтегрованому середовищі.

Особливий інтерес викликають трансформерні архітектури, що стали основою сучасних мовних моделей – BERT, FinBERT та GPT.

Їхня здатність виявляти складні контекстні залежності робить можливим інтеграцію новинного фону, аналітичних звітів і соціальних сигналів у процес фінансового прогнозування.

Проте рівень ефективності цих архітектур у задачах прогнозування ринкових часових рядів із урахуванням текстових джерел досі не досліджено достатньо.

Саме це визначає актуальність даної роботи, спрямованої на вивчення механізмів інтеграції ринкових та новинних даних у трансформерних моделях для підвищення точності фінансових прогнозів.

Об'єктом дослідження є архітектура мультимодальних нейронних мереж для прогнозування часових рядів.

Предметом дослідження є механізми інтеграції ринкових та текстових даних у трансформерних моделях (BERT, FinBERT, GPT) для підвищення ефективності фінансових прогнозів.

Метою роботи є підвищення точності прогнозування цін фінансових активів шляхом використання трансформерних архітектур (BERT, FinBERT,

GPT), здатних поєднувати ринкові часові ряди з текстовими джерелами інформації для моделювання комплексних залежностей між подіями та ринковими реакціями.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести аналіз існуючих методів прогнозування фінансових активів і визначити обмеження традиційних підходів.
2. Дослідити архітектури моделей BERT, FinBERT та GPT і визначити їх потенціал у фінансовому аналізі.
3. Розробити інтегровану модель прогнозування, яка поєднує ринкові часові ряди та текстові новинні дані.
4. Реалізувати навчання моделі на експериментальних даних і провести порівняння результатів із класичними моделями.
5. Оцінити вплив новинного фону на точність короткострокових прогнозів фінансових активів.

У процесі дослідження використано такі методи:

- аналіз часових рядів і фінансових даних;
- методи обробки природної мови (Natural Language Processing) для аналізу текстових джерел;
- архітектури глибинного навчання (LSTM, CNN, Transformer) та їх фінансові модифікації (FinBERT);
- експериментальне моделювання з використанням фреймворку PyTorch;
- кількісна оцінка результатів за метриками MAE, RMSE, MAPE.

Наукова новизна отриманих результатів полягає у розробці та дослідженні інтегрованого підходу до прогнозування цін фінансових активів на основі поєднання ринкових часових рядів і текстових новинних даних за допомогою трансформерних архітектур.

На відміну від більшості існуючих рішень, що розглядають фінансові часові ряди окремо від новинного контексту, у цій роботі запропоновано підхід,

який дозволяє об’єднати ці два типи даних у єдиній мультимодальній архітектурі.

Запропонована модель використовує трансформерні архітектури BERT, FinBERT та GPT для кодування семантичного змісту новин і їх інтеграції з ринковими ознаками, що дає змогу враховувати контекстну динаміку інформаційного середовища при прогнозуванні цін.

Крім того, у роботі проведено порівняльний експериментальний аналіз ефективності трансформерних моделей у задачах фінансового прогнозування, що дало змогу виявити вплив типу текстового представлення (BERT, FinBERT, GPT) на точність прогнозів.

Отримані результати підтвердили доцільність інтеграції текстових і ринкових даних та продемонстрували, що застосування мультимодальних трансформерних підходів може підвищити точність короткострокових фінансових прогнозів порівняно з класичними нейронними мережами.

Результати дослідження були **апробовані** під час участі у XVI міжнародній науково-практичній конференції молодих вчених «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА» (НУБіП, Київ, 2025 р.).

Магістерська робота складається зі вступу, чотирьох основних розділів, висновків, списку використаних джерел і додатків.

Загальний обсяг роботи становить 69 сторінок, з яких додатки – 13 сторінок, а список використаних джерел містить 17 найменувань.

Зміст роботи:

- перший розділ присвячений аналізу предметної області та сучасних методів прогнозування;
- другий розділ містить моделювання процесу прогнозування з використанням трансформерних моделей;
- третій розділ описує розробку інтегрованої моделі та її навчання;
- четвертий розділ включає результати експериментів і порівняльний аналіз ефективності моделей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО ПРОГНОЗУВАННЯ

1.1 Характеристика фінансових часових рядів і факторів, що впливають на їх динаміку

Фінансові часові ряди – це послідовності спостережень, що відображають зміну у часі певного ринкового показника, такого як ціна активу, обсяг торгів, валютний курс чи рівень індексу.

Вони є базовим джерелом інформації для аналітиків, трейдерів та інвесторів, оскільки дозволяють оцінювати ринкову поведінку, виявляти закономірності й будувати прогнози майбутніх змін.

Основна особливість фінансових рядів полягає у їхній високій стохастичності та нестабільності.

Ці ряди часто демонструють волатильність, тобто різкі коливання амплітуди значень навіть протягом коротких часових проміжків.

Це обумовлено багатьма зовнішніми факторами – як економічними, так і політичними чи інформаційними, які формують багатошарову структуру залежностей між даними.

Фактори, що впливають на динаміку фінансових часових рядів:

1. Економічні фактори.

Включають макроекономічні показники (ВВП, інфляція, процентні ставки, платіжний баланс, рівень безробіття), які визначають загальний стан економіки.

Наприклад, підвищення базової ставки центрального банку зазвичай спричиняє укріплення національної валюти, тоді як високий рівень інфляції може викликати зниження її курсу.

2. Політичні фактори.

До них належать зміни урядової політики, міжнародні конфлікти, санкції, вибори чи зміна політичного курсу держави.

Ці події часто викликають короткочасні, але значні коливання на фінансових ринках.

3. Інформаційні фактори.

Новини, фінансові звіти, заяви компаній або лідерів ринку, а також активність у соціальних мережах формують інформаційне поле, що суттєво впливає на поведінку учасників ринку.

Високочастотний обмін новинами створює новий рівень динаміки – новинну турбулентність, яка здатна змінити тренд ціни незалежно від фундаментальних показників.

4. Поведінкові фактори.

Це емоційні та когнітивні аспекти, які впливають на колективну поведінку інвесторів.

Наприклад, панічні продажі під час криз або масові покупки у періоди зростання можуть формувати так звані “ефекти натовпу”.

Фінансові часові ряди мають низку характеристик, які ускладнюють процес прогнозування:

- Нестаціонарність – середнє значення, дисперсія та кореляційна структура змінюються з часом.
- Нелінійність – зв’язки між змінними часто не підкоряються лінійним закономірностям.
- Гетероскедастичність – нерівномірна варіація дисперсії, що проявляється у періодах підвищеної або зниженої волатильності.
- Сезонність і циклічність – повторювані патерни, пов’язані з періодичними подіями (квартальні звіти, зміни макрополітики, святкові періоди).
- Корельованість лагів – поточні значення часто залежать від попередніх спостережень (автокореляція).

Такі властивості зумовили розвиток спеціалізованих статистичних моделей, здатних відтворювати як трендові, так і стохастичні компоненти рядів.

1.2 Методи прогнозування фінансових активів та їх еволюція

Процес прогнозування фінансових активів зазнав суттєвих змін упродовж останніх десятиліть – від класичних статистичних підходів до сучасних моделей глибинного навчання та трансформерних архітектур.

Розвиток цих методів зумовлений як ускладненням ринкових процесів, так і зростанням обсягів даних, що потребують автоматизованої обробки та інтелектуального аналізу.

Сучасні моделі мають враховувати не лише числову історію цінових коливань, але й вплив інформаційного середовища – новин, настроїв інвесторів, аналітичних оглядів, соціальних мереж.

Класичні статистичні підходи

1. Модель ARIMA (Autoregressive Integrated Moving Average)

Модель ARIMA є однією з найвідоміших і базується на поєднанні трьох компонентів:

- AR (Autoregressive) – враховує вплив попередніх значень часового ряду;
- I (Integrated) – усуває нестаціонарність шляхом диференціювання;
- MA (Moving Average) – враховує випадкові шоки або помилки попередніх прогнозів.

Загальна форма моделі ARIMA описується як:

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

де y_t- значення часового ряду у момент t, ε_t - випадкова похибка, φ_i, θ_j - коефіцієнти авторегресії та ковзного середнього, с - стала компонента.

ARIMA добре працює для коротких стаціонарних рядів, але має суттєве обмеження – лінійність. Вона не враховує складних нелінійних ефектів і не адаптується до структурних зламів у даних.

2. Модель VAR (Vector Autoregression)

Модель VAR узагальнює ARIMA для багатовимірних процесів, коли кілька змінних впливають одна на одну.

Формула VAR(p):

$$\mathbf{Y}_t = \mathbf{c} + \sum_{i=1}^{p} A_i \mathbf{Y}_{t-i} + \varepsilon_t$$

де Yt – вектор значень часових рядів, Ai – матриці коефіцієнтів взаємозв'язків, εt – вектор похибок.

Модель дає змогу враховувати взаємозв'язки між показниками, наприклад, між цінами різних валют або між ринковими індексами.

Проте, подібно до ARIMA, VAR передбачає лінійні залежності й погано працює за умов різкої зміни ринкових режимів або високої волатильності.

3. Модель GARCH (Generalized Autoregressive Conditional Heteroskedasticity)

GARCH – це клас моделей, спеціально розроблених для врахування мінливості дисперсії фінансових часових рядів.

Її загальна форма описується як:

$$\sigma_t^2 = \omega + \alpha \varepsilon_{t-1}^2 + \beta \sigma_{t-1}^2$$

де σ_t^2 – поточна дисперсія (волатильність), ϵ_t – випадкові шоки.

Модель GARCH добре описує ефект "кластеризації волатильності" – ситуацію, коли періоди високої мінливості змінюють періоди стабільності.

Завдяки цьому GARCH широко застосовується у фінансових ризик-моделях, оцінці VaR (Value at Risk) та в короткостроковому прогнозуванні волатильності.

Однак і ця модель має обмеження – вона не враховує нелінійні та текстові впливи, наприклад, вплив новин або соціальних факторів, які часто стають тригерами різких змін ціни.

Методи машинного навчання

Зі збільшенням кількості фінансових даних та ускладненням ринкових процесів почали застосовувати алгоритми машинного навчання (ML), які не вимагають чіткої математичної структури моделі.

Вони навчаються на історичних прикладах і виявляють приховані патерни у даних. До найпоширеніших методів ML у фінансовому прогнозуванні належать:

- Лінійна регресія – для базового опису тенденцій;
- Метод опорних векторів (SVM) – для класифікації трендів та станів ринку;
- Дерева рішень та ансамблеві методи (Random Forest, XGBoost, LightGBM) – для врахування нелінійних закономірностей;
- k-Nearest Neighbors (k-NN) – для пошуку локальних подібностей у даних.

Ці моделі є більш гнучкими, ніж ARIMA чи GARCH, однак не враховують часову залежність, оскільки працюють із незалежними спостереженнями. Це призвело до появи глибинних нейронних мереж, здатних аналізувати послідовності.

Глибинні нейронні мережі для часових рядів

Рекурентні нейронні мережі стали першим поколінням моделей, здатних враховувати часову структуру даних.

Мережі LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit) навчаються розпізнавати залежності між подіями, зберігаючи інформацію про попередні стани у пам’яті.

Завдяки цьому вони здатні прогнозувати тренди, цикли та коливання цін активів.

Однак, у міру збільшення довжини послідовностей, RNN (*Recurrent Neural Network*) і LSTM стикаються з проблемою затухання або вибуху градієнтів, що обмежує їх ефективність для довгострокових прогнозів.

Згорткові мережі (CNN) успішно використовуються для виявлення короткострокових патернів у даних, наприклад, локальних тенденцій або циклів у коротких часових вікнах.

У комбінації з LSTM вони формують гібридні моделі (CNN-LSTM), де CNN виділяє локальні особливості, а LSTM моделює глобальну часову структуру.

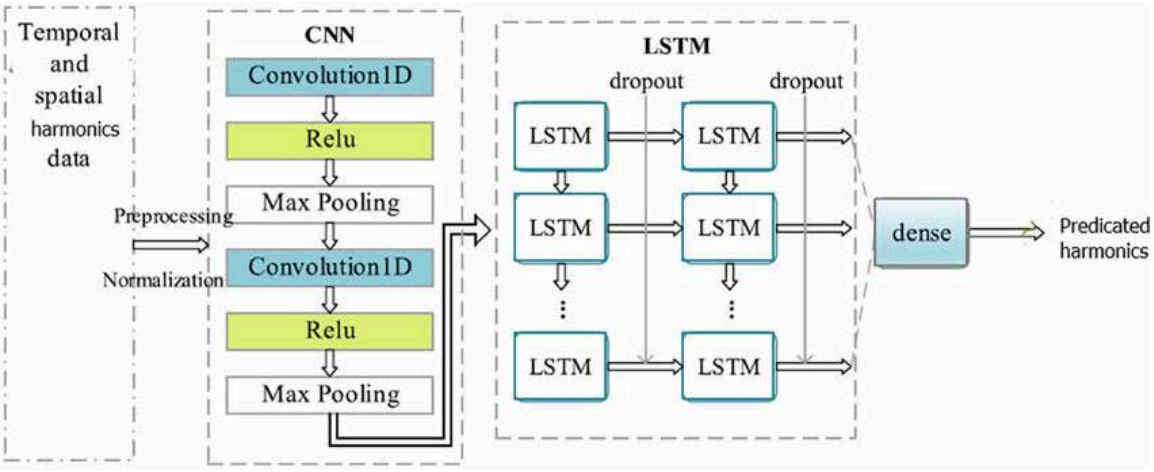


Рис. 1.1 Архітектура гібридної моделі CNN-LSTM для фінансових часових рядів.

Такі гібриди вже здатні враховувати нелінійні процеси та частково інтегрувати новинні сигнали, проте не забезпечують повноцінного контекстного аналізу.

Трансформерні архітектури у фінансовому прогнозуванні

Запропонована у 2017 році архітектура Transformer (Vaswani et al., 2017) усунула обмеження LSTM, пов’язані з послідовною обробкою даних.

Основна інновація – механізм уваги (Self-Attention), який дозволяє моделі визначати, які частини послідовності мають найбільше значення для поточного прогнозу.

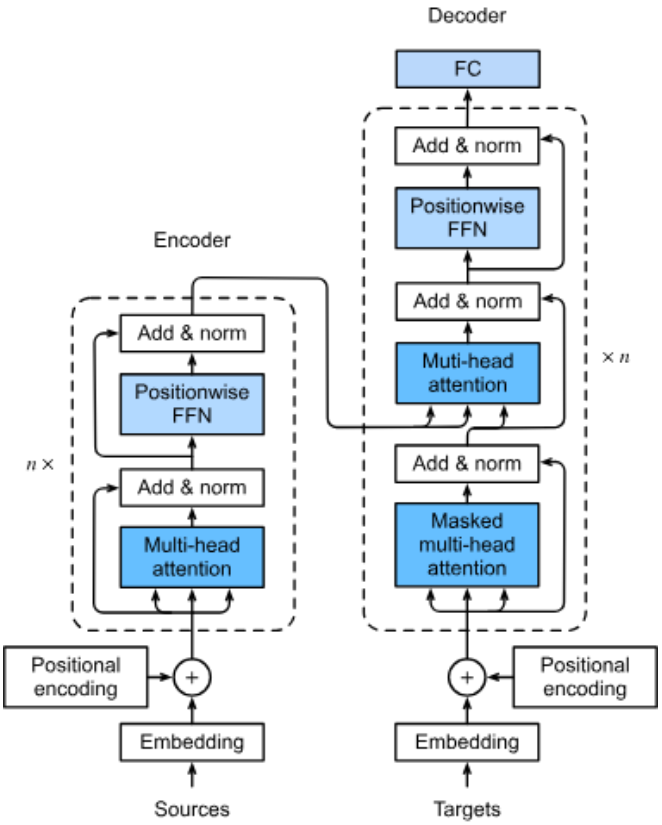


Рис. 1.2 Механізм Self-Attention у трансформерній архітектурі

Це дало змогу моделювати довгострокові залежності з високою ефективністю.

Моделі BERT (Bidirectional Encoder Representations from Transformers) і GPT (Generative Pre-trained Transformer) стали ключовими у сфері обробки природної мови (NLP). FinBERT – це адаптація BERT до фінансових текстів, навчена на економічних новинах і звітах. Вони дозволяють визначати семантичний зміст новин, тональність, контекст і приховані сигнали, які впливають на ринкові очікування.

Комбінація трансформерних мовних моделей з аналізом часових рядів створює мультимодальний підхід до прогнозування.

Такі системи об’єднують два джерела інформації:

- кількісні часові ряди (цінові, обсягові, технічні індикатори);
- текстові дані (новини, фінансові аналітики, соціальні сигнали).

Це дозволяє моделі враховувати як історичні тенденції, так і емоційно-контекстну реакцію ринку.

2 МОДЕЛЮВАННЯ ПРОЦЕСУ ПРОГНОЗУВАННЯ НА ОСНОВІ ТРАНСФОРМЕРНИХ МОДЕЛЕЙ

2.1 Формулювання задачі прогнозування та підготовка даних

Задача прогнозування формулюється як оцінка майбутньої ціни активу на основі мультимодального набору часових даних, що складається з трьох синхронізованих компонентів:

1. Ринковий часовий ряд $X_{t-w:t}$ – історія фінансових показників активу.
2. Новинний часовий ряд $N_{t-w:t}$ – історія агрегованих параметрів новин, що описують інформаційний фон.
3. Поточний новинний контекст n_t, p_t – ембеддинг і параметри останньої новини перед прогнозом.

Таким чином, прогноз обчислюється як функція:

$$\hat{y}_{t+h} = f_\Theta\left(X_{t-w:t},\ N_{t-w:t},\ n_t,\ p_t\right)$$

де $f\Theta$ – параметризована нейронна функція (модель), що інтегрує часові закономірності ринку й інформаційні впливи новин.

Ринкові дані представляються у вигляді багатовимірного часового ряду:

$$X_{t-w:t} = \{x_{t-w}, x_{t-w+1}, \ldots, x_t\}, \quad x_t \in R^{d_m}$$

Кожен вектор x_t містить набір показників, що описують стан ринку в момент часу t:

- ціна відкриття, закриття, максимум і мінімум (OHLC);
- обсяг торгів;
- похідні технічні індикатори: SMA, EMA, MACD, RSI, Bollinger Bands, волатильність;
- лагові зміни або логарифмічна дохідність.

Попередня обробка включає:

- синхронізацію часової частоти (наприклад, 1 година або 1 день);

- заповнення пропусків лінійною інтерполяцією;
- нормалізацію даних.

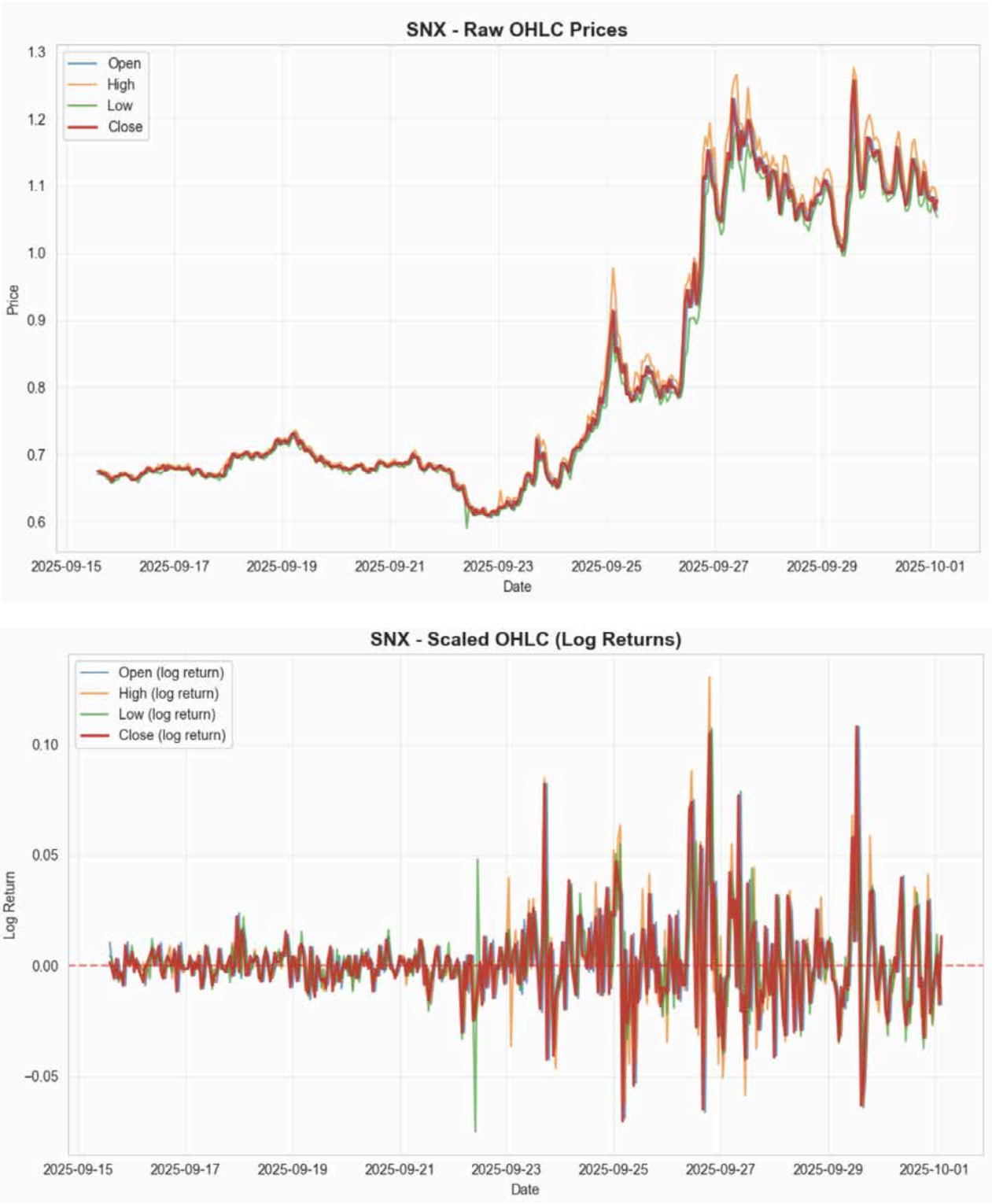


Рис. 2.1 Нормалізація цінових даних за допомогою функції log-return

Новинна історія моделюється як окремий часовий ряд параметрів, синхронізований із ринковими інтервалами. Для кожного проміжку часу

(ідентичного до ринкових спостережень) виконується агрегування властивостей усіх новин, опублікованих у цьому проміжку.

$$N_{t-w:t} = \{p_{t-w}, p_{t-w+1}, \ldots, p_t\}, \quad p_t \in R^{d_p}$$

Кожен вектор p_t містить агреговані параметри новин, зокрема:

- середня тональність (sentiment) у діапазоні [-1, 1];
- емоційна інтенсивність (intensity) – міра вираженості емоцій;
- релевантність (relevance) – відповідність новини активу чи ринку;
- довіра до джерела (trust);
- індекс інформаційної насиченості (information density) – кількість публікацій за інтервал.

Якщо протягом інтервалу було кілька новин, їхні параметри усереднюються або агрегуються з вагами за часом публікації:

$$p_t = \frac{\sum_{i=1}^{m_t} w_i \, p_i}{\sum_{i=1}^{m_t} w_i}, \quad w_i = e^{-\lambda(t-t_i)}$$

Отриманий часовий ряд $N_{t-w:t}$ описує зміну загального інформаційного настрою ринку у часовій динаміці, аналогічно до зміни цінового тренду.

Поточна новина або група найостанніших повідомлень розглядається окремо, оскільки вона безпосередньо впливає на короткостроковий прогноз і може створити ринковий шок.

Цей контекст подається у вигляді:

$$n_t = E\left(n_t^{\text{text}}\right) \in R^{d_e}, \quad p_t \in R^{d_p}$$

де E(·) – функція перетворення тексту у вектор ембеддингу, реалізована трансформерною моделлю (наприклад, FinBERT або GPT), p_t – набір параметрів тієї самої новини (тональність, важливість, тип).

Таким чином, поточний контекст поєднує семантичний зміст тексту (ембеддинг) та його метаінформаційні характеристики, що дозволяє моделі

адаптувати прогноз залежно від останніх публікацій.

Оскільки ринковий ряд $X_{t-w:t}$ та новинна історія $N_{t-w:t}$ мають однакову часову розбивку, їх можна об'єднати в єдиний мультимодальний часовий сигнал:

$$Z_{t-w:t} = \{[x_{t-w}, p_{t-w}], [x_{t-w+1}, p_{t-w+1}], \ldots, [x_t, p_t]\}$$

Таким чином, кожна точка Zi містить одночасно ринковий стан і інформаційний стан у певний момент часу.

На етапі формування навчального набору кожне спостереження має вигляд:

$$(Z_{t-w:t}, n_t, p_t, y_{t+h})$$

де y_{t+h} – реальне значення ціни через горизонт прогнозу h.

Навчальна вибірка поділяється на:

- тренувальну – для оптимізації параметрів моделі;
- валідаційну – для контролю узагальнення;
- тестову – для оцінки точності на невідомих даних.

Використовується ковзна схема валідації (rolling window validation), що імітує послідовне оновлення ринку в реальному часі.

Навчання моделі полягає у мінімізації **функції втрат**:

$$L(\Theta) = \frac{1}{N}\sum_{t=1}^{N} \rho\left(y_t - \hat{y}_t\right)$$

де $\rho(\cdot)$ – функція Penalized BerHu, що поєднує лінійні та квадратичні частини похибки:

$$\rho(e) = \begin{cases} |e|, & |e| \le \delta, \\ \dfrac{e^2 + \delta^2}{2\delta}, & |e| > \delta. \end{cases}$$

Така функція втрат робить навчання стійким до викидів, що типово для волатильних ринків.

Оптимізація виконується адаптивним методом (AdamW) з регуляризацією ваг і поступовим зменшенням швидкості навчання.

2.2 Принципи роботи моделей BERT, FinBERT та GPT

Поточний інформаційний контекст, який позначається як n_t, є одним із ключових вхідних компонентів інтегрованої моделі прогнозування.

Він представляє семантичне відображення останньої новини або групи новин, що мають безпосередній вплив на динаміку ринку.

Для його формування використовуються трансформерні архітектури – BERT, FinBERT та GPT, які здатні витягувати контекстні залежності між словами, реченнями та смисловими блоками в тексті.

На відміну від класичних моделей обробки тексту (TF-IDF, Word2Vec),

трансформери кодують значення кожного токена з урахуванням його позиції та взаємозв’язку з усіма іншими токенами в послідовності, що дає змогу відтворювати повноцінний контекст висловлювання і вловлювати тональні, емоційні та тематичні аспекти новин.

2.2.1 Архітектура BERT і механізм двонаправленої уваги

Модель BERT (Bidirectional Encoder Representations from Transformers) базується на енкодерній частині трансформера та використовує механізм двонаправленої уваги (Bidirectional Self-Attention).

Це означає, що під час обробки кожного токена модель враховує не лише попередні, а й наступні слова в реченні, завдяки чому формується повне контекстне представлення тексту.

Основні етапи роботи BERT:

1. Токенізація тексту

Вхідна новина розбивається на токени, які можуть бути цілими словами або їхніми частинами:

$$n_t^{\text{text}} \rightarrow [\text{CLS}, w_1, w_2, \ldots, w_k, \text{SEP}]$$

де [CLS] – спеціальний токен, який узагальнює значення всього тексту.

2. Позиційне кодування

Для збереження порядку токенів додаються позиційні вектори *Pi*, які повідомляють моделі про послідовність елементів у реченні.

3. Механізм самоуваги (Self-Attention)

Для кожного токена обчислюється його взаємна релевантність до всіх інших токенів у реченні за допомогою трьох матриць:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

де Q, K, V – матриці запитів, ключів і значень відповідно.

Отриманий результат визначає, які слова мають найбільший вплив на поточне.

4. Отримання векторного представлення новини

Після багатошарового проходження через блоки самоуваги вектор токена [CLS] містить узагальнену інформацію про зміст тексту:

$$E_t = \text{BERT}(n_t^{\text{text}})[\text{CLS}] \in R^{d_e}$$

Цей вектор E_t використовується як частина поточного контексту n_t.

2.2.2 FinBERT – спеціалізована модель для фінансових текстів

FinBERT є модифікацією базової архітектури BERT, адаптованою для аналізу фінансових текстів – корпоративних звітів, новин, аналітики, біржових публікацій.

Основні відмінності FinBERT:

1. Доменно-специфічне навчання.

Модель додатково тренується на фінансових корпусах (Reuters, SEC Filings, Financial PhraseBank), тому краще розпізнає фінансову термінологію, скорочення, контекст ризиків та оцінок.

2. Вбудоване визначення тональності (Sentiment Head).

FinBERT оцінює текст не лише семантично, але й за емоційним знаком:

$$s_t = \text{softmax}(W_s E_t + b_s)$$

де $s \in [-1,1]$ – рівень позитивності або негативності новини.

Це значення використовується як один з елементів параметрів *pt*.

3. Контекстна агрегація тематик.

Вектор FinBERT відображає загальний "настрій" ринку, зокрема інформаційні тригери (страх, оптимізм, нейтральність).

Таким чином, результат роботи FinBERT складається з двох частин:

- вектор E_t^{fin} семантичне представлення змісту;
- набір $p_t = [s_t, \text{relevance}, \text{trust}, \text{intensity}]$ – параметри новини.

2.2.3 GPT – генеративний контекстний модуль

Модель GPT (Generative Pre-trained Transformer) базується лише на декодерній частині трансформера та використовує односпрямовану увагу (Causal Attention).

Це означає, що кожне слово враховує лише попередні токени, що дозволяє моделі формувати послідовне уявлення контексту.

У нашому підході GPT використовується не для генерації тексту, а для додаткового контекстного кодування, яке вловлює приховані зв’язки між словами, що можуть відображати майбутні реакції ринку. Таким чином формується генеративний вектор впливу:

$$E_t^{\text{gpt}} = \text{GPT}(n_t^{\text{text}})$$

який доповнює контекст n_t, отриманий із BERT/FinBERT.

Після об’єднання всіх джерел формується остаточний вектор поточного контексту:

$$n_t = [E_t^{\text{bert}}, E_t^{\text{fin}}, E_t^{\text{gpt}}]$$

а також супровідний набір параметрів p_t, який містить числові характеристики тексту:

$$p_t = [s_t, \text{trust}, \text{intensity}, \text{relevance}]$$

2.2.4 Інтеграція поточного контексту в модель прогнозування

Після формування векторів n_t і p_t вони надходять до контекстного модуля інтеграції, де перетворюються у компактне представлення через невелику нейронну мережу:

$$h_t^{\text{context}} = \sigma(W_n n_t + W_p p_t + b)$$

де σ(·) – активаційна функція (GELU або tanh), а $h_t^{context}$ – фінальний вектор контекстного впливу.

Цей вектор далі передається в основну трансформерну модель, де він модулює прогноз через механізм уваги або через ф'юзер (Gating).

Таким чином:

- BERT формує базове двонаправлене представлення контексту;
- FinBERT забезпечує спеціалізоване фінансове розуміння і визначає тональність;
- GPT підсилює контекст за рахунок причинно-послідовного моделювання.

А їхня комбінація створює узагальнений вектор поточного інформаційного впливу, який, разом із параметрами p_t, визначає інформаційний фон ринку в момент прогнозу.

2.3 Математична модель і алгоритмічна схема прогнозування

Розроблена система прогнозування є мультимодальною ієрархічною моделлю, у якій поєднуються дані з двох різних джерел – ринкових часових рядів і новинного інформаційного контексту.

Основна ідея полягає в тому, що ринкові тенденції та інформаційні сигнали з новин є взаємодоповнювальними, але різної природи, тому їх необхідно обробляти окремими підмодулями з подальшим інтеграційним узгодженням (fusion).

Формально модель описується як функція:

$$\hat{y}_{t+h} = F_{\Theta}\big(f_m(X_{t-w:t}), f_n(N_{t-w:t}), f_c(n_t, p_t)\big)$$

де:

- f_m – модуль обробки ринкових часових рядів;
- f_n – модуль обробки історії новинних параметрів;
- f_c – модуль поточного контексту;
- F_{Θ} – функція інтеграційного рівня, що поєднує результати попередніх модулів для прогнозу.

Кожен підмодуль виконує свою специфічну роль у формуванні представлень (feature representations), а інтеграційний блок узгоджує їх на рівні спільного латентного простору.

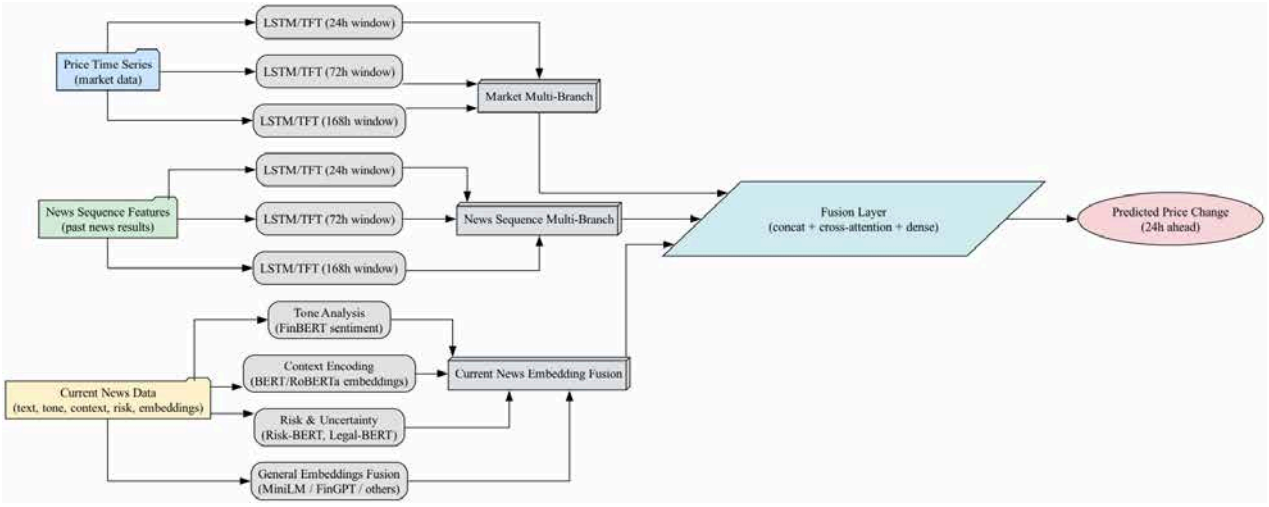


Рис. 2.2 Узагальнена архітектура мультимодальної моделі прогнозування

Модель складається з трьох функціональних рівнів:

1. Ринковий рівень – аналізує динаміку фінансових показників і виявляє коротко- та довгострокові закономірності.

2. Новинний рівень – відстежує зміну параметрів новинного фону (тональність, довіра, інтенсивність) у часі.

3. Контекстний рівень – враховує останню новину або групу новин, які безпосередньо впливають на ринок у момент прогнозу.

Результати кожного рівня подаються у вигляді внутрішніх (латентних) векторів, що описують стан відповідної модальності, після чого проходять інтеграційний етап – механізм злиття (fusion). На цьому етапі модель узгоджує різні типи інформації та формує спільне уявлення про ринкову ситуацію.

Ринковий рівень виконує функцію виявлення патернів, трендів і залежностей у часових рядах. Він поєднує аналіз локальних змін (наприклад, короткі імпульси, коливання) з моделюванням довгострокових залежностей, які відображають фундаментальні рухи цін. Отримане представлення характеризує поточний стан ринку та його інерцію.

Новинна історія розглядається як окремий часовий ряд параметрів, синхронізований із ринковими спостереженнями. Він відображає зміну емоційного, тематичного та інформаційного стану ринку. Модель на цьому рівні відстежує поступові зміни тональності новин, загальну насиченість інформаційного поля та динаміку довіри до джерел. Таким чином, формується узагальнений показник впливу новинного середовища на ринок у часі.

Поточний контекст враховує найсвіжіші новини, що можуть миттєво впливати на ринкові очікування або спричиняти короткострокові коливання. Цей рівень використовує векторні представлення змісту тексту (ембеддинги, отримані трансформерними моделями) та числові параметри новини (тональність, достовірність, інтенсивність). Отриманий контекстний вектор відображає прямий інформаційний імпульс і передається на інтеграційний рівень.

На інтеграційному рівні відбувається поєднання ознак з усіх модальностей – ринкової, історичної та контекстної. Для цього використовується адаптивний механізм злиття, який визначає, яка інформація має більшу вагу у поточному прогнозі:

$$h_f = \alpha_m h_m + \alpha_n h_n + \alpha_c h_c, \quad \alpha_i = \text{softmax}(w_i^T h_i)$$

Алгоритм роботи системи прогнозування узагальнено складається з таких етапів:

1. Отримання ринкових і новинних даних за синхронізованою часовою шкалою.
2. Обробка ринкових даних і новинної історії в окремих підмодулях.
3. Формування поточного контексту на основі останньої новини.
4. Інтеграція результатів усіх модулів на спільному рівні злиття.
5. Генерація прогнозу та порівняння з фактичним результатом.
6. Оновлення параметрів моделі відповідно до похибки прогнозу.

Цей процес повторюється ітеративно, що забезпечує безперервне навчання моделі та її здатність адаптуватися до змін ринкового середовища.

3 РОЗРОБКА ІНТЕГРОВАНОЇ МОДЕЛІ ТА ЇЇ НАВЧАННЯ

3.1 Архітектура моделі Integrated Hierarchical Model

Модель Integrated Hierarchical Model v8.4 реалізує мультимодальне прогнозування часових рядів на основі двох основних джерел часових даних:

- ринковий часовий ряд (Market, x_market),
- часовий ряд параметрів новин (NewsHistory, x_news_history), а також поточного новинного контексту, який подається у вигляді:
- векторів ембеддингів новин (один або кілька embedding'ів з трансформерів BERT / FinBERT / GPT),
- вектора параметрів новини (news_params).

На виході модель формує прогноз часових рядів із заданим горизонтом (horizon, кількість кроків уперед) та кількістю прогнозованих змінних (out_dim).

Архітектура складається з таких великих компонентів:

- ParameterGroup для ринку (Market) – багатовіконний, багатовітковий блок обробки ринкових даних.
- ParameterGroup для новинної історії (NewsHistory) – аналогічний блок для параметричного новинного ряду.
- TS fusion (GatingMLPFusion) – злиття ринкового та новинно-історичного представлень у єдиний часовий сигнал.
- NewsFusion + DropNews – об'єднання новинних embedding'ів і параметрів в один ознаковий вектор
- Cross-Modal Attention (pre / post / both) – опціональна перехресна увага між часовими ознаками та новинами.
- Декодер часових рядів (Parallel / AR-LSTM / AR-Transformer) – генерація прогнозу з TS-ознакового простору.
- News-based head + Confidence & Trust gating – паралельний прогноз із новинного шляху і вихідне злиття.
- Вбудована підтримка функції втрат (в т.ч. Penalized BerHu) і моніторинг ваг / ентропій / температур τ.

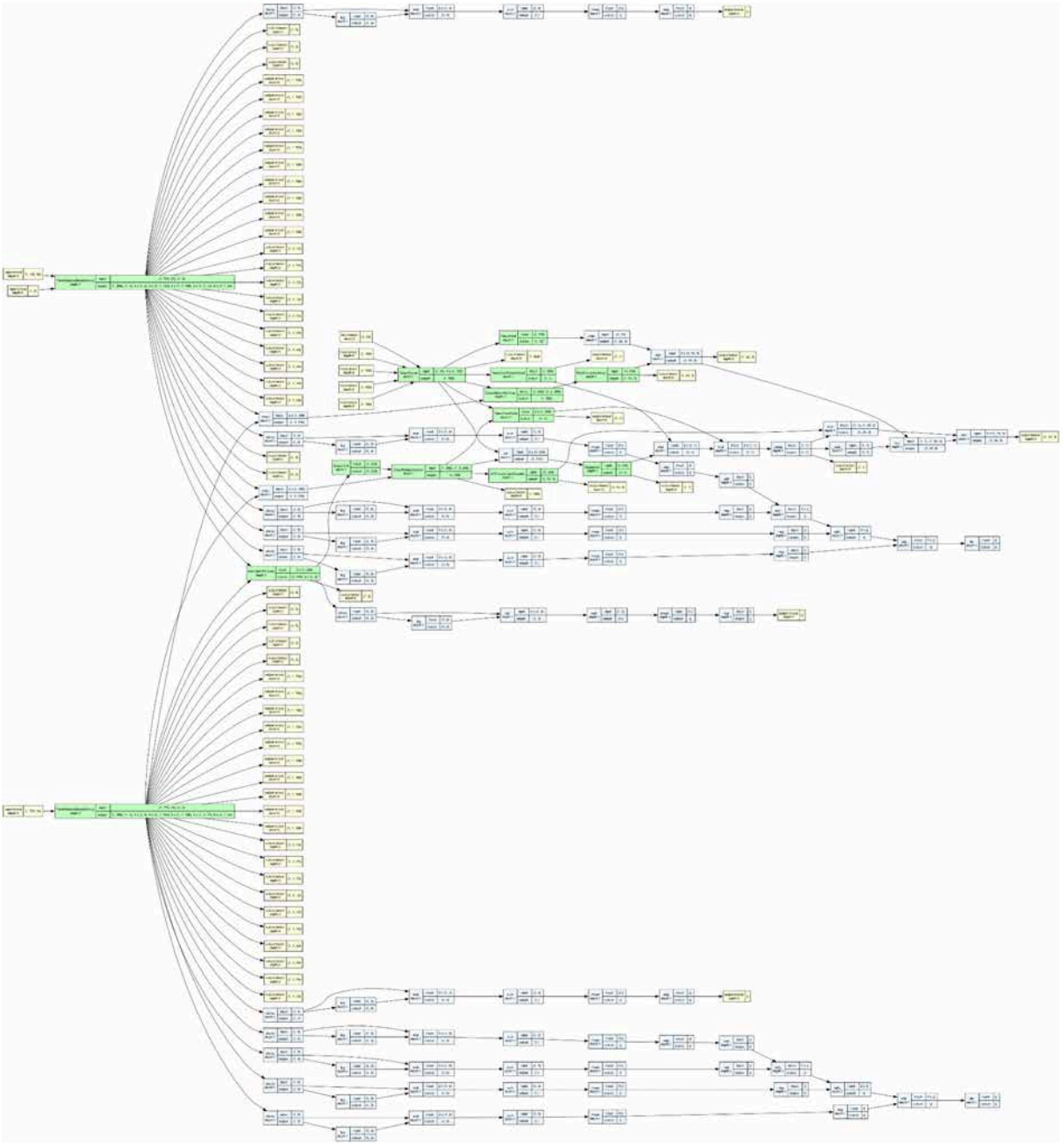


Рис. 3.1 Архітектура моделі Integrated Hierarchical Model v8.4

3.1.1 ParameterGroup для ринку та новинної історії

Для кожного джерела часових рядів (ринок і новинна історія) використовується одна і та сама узагальнена структура – TimeWindowBranchGroup.

- Вона працює так: на вхід подається тензор розміру [B, T, C], де: B – розмір батчу, T – довжина часового вікна, C – кількість ознак (ринкових або новинних параметрів);

- для кожного джерела задається набір вікон по часу (market_windows, news_hist_windows), наприклад: 24, 72, 168, 720 останніх кроків;
- для кожного вікна використовується набір гілок (branches), створених фабриками: market_branch_factories, news_hist_branch_factories.

Ці гілки – це різні моделі (наприклад, CNN+LSTM, LSTM+attention, спрощений TFT, трансформер-ветка тощо);

- кожна гілка для конкретного вікна повертає або послідовність ознак [B, T', H], або вектор [B, H]. Якщо це послідовність – до неї може застосовуватись MultiHeadAttentionPooling: багатоголова увага по часовій осі, яка агрегує її в вектор [B, H] і повертає карту уваги по часу;
- отримані вектори з усіх гілок одного вікна передаються в GatingMLPFusion: невелика MLP мережа обчислює ваги для кожної гілки (через softmax з температурою τ), після чого вектори зважено сумуються в один вектор для вікна;
- далі вектори всіх вікон (короткі, середні, довгі горизонти) знову агрегуються ще одним GatingMLPFusion уже на рівні вікон. Як результат, для одного джерела (ринку або новинної історії) отримуємо:
  - fused – підсумковий вектор ознак розмірності [B, H];
  - window_weights – ваги по вікнах;
  - branch_weights_per_window – ваги по гілках у кожному вікні;
  - time_attn_maps – карти часової уваги (якщо включено attention-пулінг).

Таким чином:

- self.market = TimeWindowBranchGroup(...) – обробка ринкових даних;
- self.news_hist = TimeWindowBranchGroup(...) – обробка новинного параметричного ряду.

3.1.2 Новинний шлях: NewsFusion, DropNews, Confidence, Trust

Паралельно з часовою частиною працює новинний шлях. Він включає кілька підмодулів:

NewsFusion

Модуль NewsFusion приймає:

- список ембеддингів новин news_embeddings – один або декілька векторів [B, E_i] (наприклад, з BERT, FinBERT, GPT);
- вектор параметрів новини news_params [B, P] (тональність, інтенсивність, довіра тощо).

Далі він:

- проєктує кожен embedding у спільний простір розмірності hidden через Linear + GELU + LayerNorm;
- опційно проєктує параметри новин у той самий простір;
- конкатенує всі проєктовані вектори та пропускає їх через MLP, формуючи узагальнений вектор news_feat [B, H].

DropNews

Усередині NewsFusion реалізовано регуляризацію DropNews:

з деякою ймовірністю (drop_news_p) модуль у режимі навчання повертає нульовий вектор замість реального news_feat.

Це змушує модель не покладатися надмірно на новинну гілку і вчитися використовувати ринкову інформацію навіть за “відсутності” новин.

NewsConfidenceHead

Модуль NewsConfidenceHead приймає news_feat і видає скалярну оцінку впевненості в новинній інформації у діапазоні [0, 1]. Це окрема невелика мережа з LayerNorm, повнозв’язних шарів і сигмоїди.

NewsTrustGate

Модуль NewsTrustGate приймає одночасно:

- news_feat – новинні ознаки,
- ts_feat – часові (ринок+історія новин).

Він поєднує їх, подає у MLP і повертає ще один скаляр [0,1] – ступінь довіри до новин з огляду на поточний стан ринку.

У підсумку, новинний шлях дає:

- сам прогноз із новин (y_news),
- ознаковий вектор news_feat,
- два скаляри – confidence і trust, які використовуються на етапі вихідного злиття.

3.1.3 Cross-Modal Attention (опціонально)

Модель підтримує опціональний блок CrossModalAttention, який може працювати на двох стадіях:

- pre – до часової декодувальної голови,
- post – як коректор новинного прогнозу,
- both – обидва варіанти одночасно.

У цьому блоці:

- query – ознаки з однієї модальності (наприклад, ts_feat або news_feat),
- context – стек ознак з ринку й новинної історії [B, 2, H].

Використовується MultiheadAttention з власною температурою τ і пост-обробкою через LayerNorm + невеликий MLP. У режимі навчання температура також аннелюється, поступово роблячи увагу більш “гострою”.

3.1.4 Декодери часових рядів

Для перетворення інтегрованого часово-новинного ознакового вектора ts_feat у прогноз модель підтримує три варіанти декодера (decoder_type):

- "parallel" → ParallelForecastHead

простий лінійний шар, який відразу прогнозує весь горизонт [B, H, out_dim] від одного вектора ts_feat.

- "ar_lstm" → ARLSTMDecoder

авторегресивний декодер на базі LSTM, на кожному кроці використовує попередній прогноз (або teacher forcing) і контекст ts_feat, підтримує обмеження виходу (lower_bound, upper_bound).

- "ar_transformer" → ARTransformerDecoder

авторегресивний трансформер-декодер з позиційними embedding’ами, на кожному кроці дивиться на попередні кроки в послідовності прогнозу та на контекст із часових ознак.

4.1.5 Вихід моделі та вбудована функція втрат

Метод forward повертає словник, у якому є:

- y – фінальний прогноз [B, H, out_dim];
- loss – опціонально, якщо передано criterion і targets (можна напряму використовувати Penalized BerHu всередині моделі);
- проміжні величини для аналізу / інтерпретації:
  - ваги вікон (market_window_weights, news_hist_window_weights),
  - ваги гілок, часові карти уваги,
  - ваги джерел (ts_src_weights),
  - температури τ для різних механізмів,
  - новинні ознаки news_feat та ін.

Це дозволяє не лише отримувати прогноз, а й аналізувати, які саме вікна, гілки та модальності були найважливіші в конкретний момент.

3.2 Модулі обробки ринкових і новинних даних

Модулі обробки часових даних є центральною частиною архітектури Integrated Hierarchical Model.

Вони відповідають за вилучення закономірностей у часових рядах ринку та динаміці параметрів новинного фону.

Усі часові обчислення побудовані на базовій структурі TimeWindowBranchGroup, яка поєднує багатовіконний і багатовітковий принципи обробки.

Компонент TimeWindowBranchGroup реалізує концепцію multi-window / multi-branch processing, тобто для кожного джерела часових рядів створюється набір вікон різної довжини, а в межах кожного вікна – кілька паралельних гілок (branch factories) із різними типами часових енкодерів.

Кожна гілка отримує зріз часових даних певної довжини [B, T, C] і формує або часову послідовність ознак [B, T', H], або агрегований вектор [B, H].

Усі гілки однієї групи мають незалежні параметри, але спільну структуру інтерфейсу: вони реалізують метод forward, який повертає результат і, за потреби, карту уваги. Результати гілок передаються до механізму GatingMLPFusion, який виконує динамічне злиття з урахуванням їх вагомості.

Ваги обчислюються невеликим MLP через softmax і контроль температури τ. Завдяки цьому модель навчається розподіляти вплив між короткостроковими і довгостроковими закономірностями.

Результати обчислень кожного вікна потім узгоджуються між собою другим рівнем GatingMLPFusion.

Це формує підсумковий вектор ознак для джерела (ринкового чи новинного),

а також набір допоміжних параметрів: ваги вікон, ваги гілок і карти часової уваги.

У підсумку блок TimeWindowBranchGroup повертає структуру:

- fused – агрегований вектор ознак [B, H];
- window_weights – ваги по часових вікнах;
- branch_weights_per_window – ваги по гілках у кожному вікні;
- time_attn_maps – карти часової уваги (якщо активовано AttentionPooling).

Цей механізм використовується двічі – для ринкового ряду (self.market) і новинного ряду (self.news_hist).

Типи часових гілок

Модель здатна обробляти паралельно велику кількість гілок що довзволяє використовувати різні підходи одночасно та концентруватися на найбільш ефективних. Гілки реалізації використовуються в межах TimeWindowBranchGroup.

Основні з них:

- LSTMBranch – класична рекурентна гілка, що використовує LSTM для моделювання часової послідовності. Підходить для виявлення плавних трендів та середньострокових залежностей.
- TransformerBranch – базується на багатоголовій самоувазі (Multihead Self-Attention) і дозволяє моделі навчати взаємозв’язки між усіма кроками у вікні, зберігаючи інформацію про порядок за рахунок позиційного кодування. Цей варіант добре працює для довгострокових зв’язків.
- CNNLSTMBranch – комбінує згорткові шари (Conv1D) для локального згладження і LSTM для часової інтеграції. Використовується для коротких або середніх вікон.
- LightTFTBranch – спрощена версія Temporal Fusion Transformer із лінійним шаром проєкції і шаром уваги. Дає кращу продуктивність при малих даних.

Для кожної гілки може використовуватися AttentionPooling – механізм, який агрегує часову послідовність у вектор через багатоголову увагу. Це дозволяє адаптивно вибирати, які часові моменти мають найбільшу вагу.

TS Fusion

Після обробки обох джерел (market, news_history) їхні вектори [B, H] об’єднуються через модуль ts_sources_fuser – реалізацію GatingMLPFusion для двох джерел.

Він навчається визначати, яке джерело (ринок чи новинна історія) є більш інформативним у поточний момент. Softmax-ваги (ts_src_weights) контролюються температурою τ, яка зменшується протягом навчання (механізм annealing).

Це дозволяє моделі спочатку досліджувати різні співвідношення між модальностями, а згодом концентруватися на найстабільніших джерелах.

Результатом є часовий вектор ts_feat, що узагальнює спільну динаміку ринку та новинної історії, і виступає базовим представленням для наступних рівнів моделі.

NewsFusion і модулі новинного контексту

Новинний контекст формується окремим модулем NewsFusion.

Його вхід – список ембеддингів новин (news_embeddings) і вектор параметрів (news_params). Кожен ембеддинг проєктується у спільний простір hidden через послідовність Linear → GELU → LayerNorm. Після цього усі ембеддинги об’єднуються з параметрами і проходять через багатошаровий перцептрон.

Результат – вектор news_feat, який описує поточну новину на семантичному рівні.

Для регуляризації реалізовано DropNews: під час навчання з певною ймовірністю новинна ознака занулюється. Це підвищує стійкість моделі до відсутності або низької якості новин.

Додатково використовуються два модулі контролю якості сигналу:

- NewsConfidenceHead – оцінює впевненість у новинній інформації (0–1), реалізований як MLP з нормалізацією і сигмоїдою.
- NewsTrustGate – обчислює узгодженість між новинами та ринковим станом, використовуючи одночасно news_feat і ts_feat.

Результатом є три сигнали: news_feat, confidence, trust, які використовуються під час формування фінального прогнозу.

3.3 Механізм злиття ознак і прогнозувальний блок

Після обробки часових і новинних потоків модель переходить до інтеграційного рівня, який об’єднує усі модальні представлення в єдине прогнозне рішення. Ключова логіка цього рівня визначається модулями GatingMLPFusion, CrossModalAttention, а також Decoder Head, що перетворює інтегроване представлення в реальний прогноз.

Інтеграційний шар (Fusion Layer)

Основне завдання Fusion Layer – узгодити часову ознакову репрезентацію ts_feat із поточним новинним контекстом news_feat.

Для цього використовується комбінація трьох скалярних коефіцієнтів: base_alpha, confidence і trust. Їх добуток утворює загальний коефіцієнт злиття

$$\alpha = clip(base\_alpha \cdot confidence \cdot trust)$$

який задає баланс між часовим і новинним прогнозом.

Фінальний прогноз обчислюється як опукла комбінація:

$$y = \alpha \cdot y_{ts} + (1 - \alpha) y_{news}$$

Таким чином, модель динамічно змінює вагу впливу новинного контексту в залежності від ринкової ситуації.

Cross-Modal Attention

Опціонально активується механізм CrossModalAttention, який може бути використаний до або після етапу прогнозування (pre, post або both). Він дозволяє новинним ознакам безпосередньо впливати на часові, використовуючи механізм багатоголової уваги.

Вхідні пари (query, context) формуються з ts_feat і [mkt_fused, nh_fused].

Температура τ визначає "гостроту" уваги і зменшується під час навчання, що забезпечує поступовий перехід від глобального до локального фокусування.

Декодери

Для перетворення інтегрованого ознакового вектора в прогноз модель підтримує три типи декодерів (decoder_type):

- Parallel Forecast Head – паралельне прогнозування всього горизонту;
- ARLSTMDecoder – покроковий авторегресійний прогноз із LSTM;
- ARTransformerDecoder – трансформер-декодер із позиційним кодуванням.

Вибір декодера визначається типом задачі:

- для коротких горизонтів – паралельний,
- для середніх і довгих – авторегресійний.

Функція втрат із штрафами (Penalized BerHu Loss)

У фінансовому режимі модель використовує Penalized BerHu Loss, яка складається з двох логічних частин:

1. базова BerHu-втрата за модулем помилки;
2. додаткові штрафи за "неправильну" поведінку ціни:
   - неправильний напрямок руху,

- вихід за допустимий діапазон (overshoot),
- факт виходу за межі (out of bounds, навіть якщо перевищення маленьке).

Спочатку для кожного моменту часу й компоненти прогнозу обчислюється помилка:

$$e_t = \hat{y}_t - y_t$$

У фінансовому режимі використовується фіксований поріг δ (параметр delta), і базова частина втрат визначається як:

$$L_{\text{BerHu}}(e_t) = \begin{cases} |e_t|, & |e_t| \le \delta, \\ \dfrac{e_t^2 + \delta^2}{2\delta}, & |e_t| > \delta, \end{cases}$$

- для невеликих помилок ($|e| \le \delta$) працює лінійний режим – як L1;
- для великих помилок ($|e| > \delta$) – квадратичний режим, який сильніше карає великі промахи.

Середнє значення по всім точкам дає базову частину втрат:

$$L_{\text{base}} = \text{mean}\big(L_{\text{BerHu}}(|\hat{y} - y|)\big)$$

Окрім величини помилки, у фінансових задачах важливий знак зміни ціни. Функція втрат додає штраф, якщо модель передбачила неправильний напрямок руху (вгору/вниз).

Для цього порівнюється знак різниць між сусідніми кроками:

$$\text{sign}_{\text{true}} = \text{sign}(y_t - y_{t-1}), \quad \text{sign}_{\text{pred}} = \text{sign}(\hat{y}_t - \hat{y}_{t-1})$$

де $sign_{true}$ – справжній напрямок, а $sign_{pred}$ – передбачений напрямок.

Якщо добуток $sign_{true}$ * $sign_{pred}$ < 0, це означає, що прогноз “попав у протилежний бік” (ринок ріс, модель сказала падіння або навпаки). Тоді додається штраф:

$$L_{\text{dir}} = \lambda_{\text{dir}} \cdot \text{mean}\big(\mathbf{1}\{\text{sign}_{\text{pred}} \cdot \text{sign}_{\text{true}} < 0\}\big)$$

де λ_{dir} – коефіцієнт direction_penalty.

Цей член змушує модель не просто наближати числове значення, а правильно вгадувати напрямок тренду.

Для кожного моменту часу модель може отримувати допустимий діапазон ціни:

- нижня межа lower_bound,
- верхня межа upper_bound.

Якщо прогноз виходить за ці межі, обчислюються величини превищення:

$$\text{over}_{\text{upper}} = \max(0,\ \hat{y} - \text{upper\_bound}), \quad \text{under}_{\text{lower}} = \max(0,\ \text{lower\_bound} - \hat{y})$$

де $over_{upper}$ – перевищення верхньої межі, а $under_{lower}$ – вихід нижче нижньої межі.

Штраф за overshoot:

$$L_{\text{over}} = \lambda_{\text{over}} \cdot \text{mean}\big(\text{over}_{\text{upper}} + \text{under}_{\text{lower}}\big)$$

де λ_{over} – коефіцієнт overshoot_penalty.

Цей термін карає величину виходу за межі: чим далі прогноз за діапазоном – тим більший штраф.

Окремо враховується сам факт, що прогноз взагалі вийшов за [lower_bound, upper_bound], навіть якщо вихід невеликий.

Створюється бінарна маска і відповідний штраф:

$$\text{out\_of\_bounds} = \mathbf{1}\{\hat{y} > \text{upper\_bound} \vee \hat{y} < \text{lower\_bound}\}$$

$$L_{\text{bound}} = \lambda_{\text{bound}} \cdot \text{mean}\big(\text{out\_of\_bounds}\big)$$

де λ_{bound} – коефіцієнт bound_penalty.

Тобто:

- overshoot штрафує за наскільки сильно прогноз виліз за рамки;
- bound – за сам факт порушення діапазону, навіть якщо воно маленьке.

У фінансовому режимі повна функція втрат має вигляд:

$$L_{\text{total}} = \text{mean}\big(L_{\text{BerHu}}(|\hat{y}-y|)\big)+\lambda_{\text{dir}}\,\mathbb{E}[\text{wrong\_sign}]+\lambda_{\text{over}}\,\mathbb{E}[\text{overshoot}]+\lambda_{\text{bound}}\,\mathbb{E}[\text{out\_of\_bounds}]$$

де:

- E[wrong_sign] – частка кроків, де напрямок змін передбачено неправильно;
- E[overshoot] – середній розмір перевищення межі;
- E[out_of_bounds] – частка прогнозів, що вийшли за діапазон.

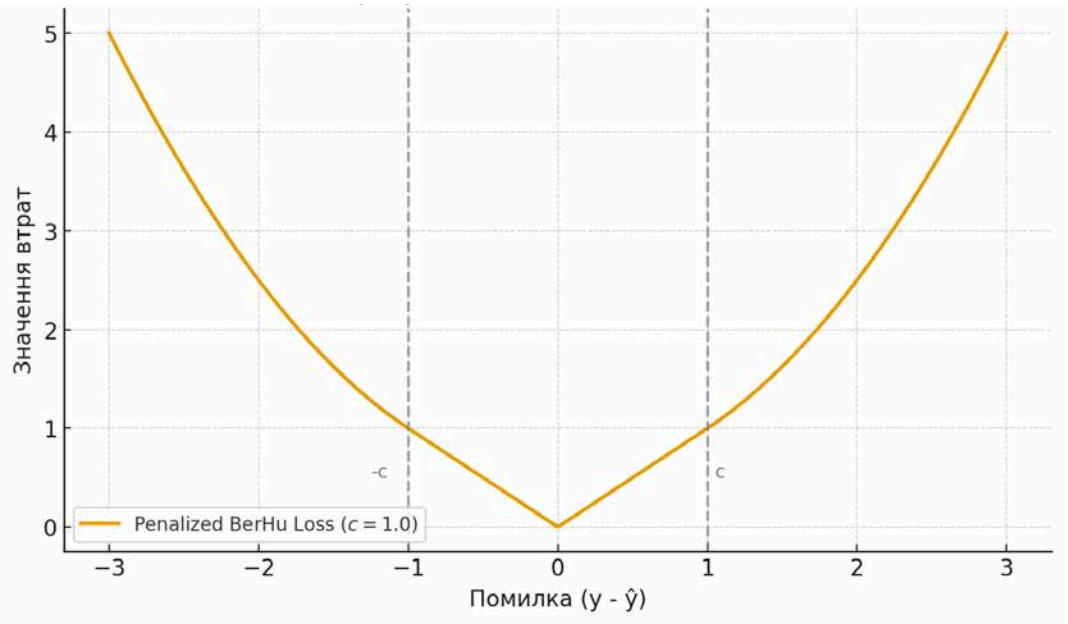


Рис. 3.2 Графік Penalized BerHu Loss

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Умови навчання та параметри експериментів

Процес навчання моделі Integrated Hierarchical Model побудовано як багатофазову систему адаптивного мультимодального навчання. Метою цього етапу є забезпечення узгодженого розвитку часових і новинних підсистем, стабільна робота механізмів злиття (fusion) та формування каузально коректного прогнозу, що одночасно враховує динаміку ринку й поточний інформаційний фон.

Навчання проводилось у середовищі глибокого навчання з апаратним прискоренням (GPU).

Застосовувався режим Automatic Mixed Precision (AMP), що дозволяє комбінувати обчислення у форматах FP16 / bfloat16 і FP32 без втрати точності, зменшуючи навантаження на пам’ять і прискорюючи виконання.

Архітектурні гіперпараметри моделі

Ця група параметрів визначає структуру IHM: кількість вхідних ознак, часові вікна, розмірність прихованих шарів, використання міжмодальної уваги та регуляризацію.

Таблиця 4.1

Архітектурні гіперпараметри моделі

Параметр	Опис	Значення
hidden	Розмірність прихованого представлення	256
horizon	Горизонт прогнозу (кількість майбутніх кроків)	24
out_dim	Кількість вихідних змінних	3
market_input_dim	Кількість ринкових ознак	30
market_windows	Довжини часових вікон для ринку	[30, 60, 120]
news_hist_input_dim	Кількість параметрів новинної історії	10
news_hist_windows	Вікна для новинної історії	[30, 60]

Продовження таблиці 4.1

Параметр	Опис	Значення
news_embed_dims	Розміри ембеддингів новин (BERT/FinBERT/GPT)	[728, 728, 728]
news_params_dim	Розмірність параметрів поточної новини	10
use_cross_modal	Використання міжмодальної уваги	True
cross_modal_stage	Етапи застосування уваги	"both"
n_heads_cross	Кількість голів у Cross-Modal Attention	8
dropout	Імовірність Dropout	0.2
drop_news_p	Початкова ймовірність DropNews	0.3

Основні гіперпараметри навчання

Параметри, що безпосередньо впливають на процес оптимізації, швидкість збіжності та стабільність навчання.

Таблиця 4.2

Основні гіперпараметри навчання

Параметр	Опис	Значення
epochs	Кількість епох навчання	50
lr	Початкова швидкість навчання	3e−4
weight_decay	L2-регуляризація	1e−4
scheduler	Тип планувальника швидкості	"cosine"
warmup_epochs	Епохи розігріву	3
min_lr	Мінімальне значення Learning Rate	1e−6
amp	Використання змішаних обчислень (FP16/FP32)	True
grad_clip_norm	Максимальна норма градієнта	1.0
tf_ratio	Початкове значення teacher forcing	1.0
tf_ratio_min	Мінімальне значення teacher forcing	0.0
tf_ratio_decay_epochs	Тривалість зменшення teacher forcing	40

Продовження таблиці 4.2

Параметр	Опис	Значення
use_ema	Використання EMA ваг (експоненційне усереднення)	True
ema_decay	Коефіцієнт згладжування EMA	0.999
seed	Фіксоване зерно випадковості	42
device	Тип пристрою для обчислень	auto ("cuda" / "mps" / "cpu")

Гіперпараметри функції втрат

Функція втрат поєднує модифіковану BerHu (Reverse Huber) з адаптивними штрафами, що відображають реальні ринкові обмеження – напрям руху ціни, перевищення меж та факт виходу за діапазон.

Таблиця 4.3

Гіперпараметри функції втрат

Параметр	Опис	Значення
mode	Режим роботи функції втрат	"finance"
delta	Поріг переходу між L1 і L2	0.5
direction_penalty	Штраф за неправильний напрямок зміни ціни	2.0
overshoot_penalty	Штраф за перевищення меж (overshoot)	1.5
bound_penalty	Штраф за вихід за допустимі межі	1.5

Сценарії експериментів

Для дослідження ефективності окремих модальностей було проведено кілька навчальних сценаріїв:

- Market-only: модель отримує лише часові ринкові дані.
- Market + NewsHistory: додається часовий ряд агрегованих параметрів новин.
- Full multimodal: активуються всі модальності, включно з поточним новинним контекстом.

- Full multimodal + Inverse α: тестовий режим із інверсією ваг злиття для перевірки стабільності балансу між модальностями.

Порівняння результатів проводилось за середніми значеннями MAE, RMSE і MAPE, а також за поведінкою коефіцієнта α (ступінь впливу новинного контексту).

Такі експерименти дозволили кількісно оцінити, як саме новини покращують точність прогнозів і як змінюється роль контексту у фазах волатильності ринку.

Відтворюваність експериментів

Для забезпечення коректного порівняння результатів навчання було встановлено фіксоване початкове зерно випадковості (random seed) для всіх використовуваних генераторів випадкових чисел: seed = 42.

Це зерно передавалось одночасно у:

- генератор PyTorch,
- генератор NumPy,
- системний генератор Python,
- а також у CUDA-графічні генератори.

Завдяки цьому гарантовано однакове початкове ініціалізування ваг моделі, однаковий розподіл батчів у даталоадері та синхронну поведінку Dropout і DropNews між запусками.

Такий підхід дозволив забезпечити повну відтворюваність результатів між серіями навчань та об’єктивність порівнянь між моделями.

4.2 Експериментальний аналіз точності прогнозування та впливу новинного фону

Метою цього етапу дослідження є оцінка точності прогнозів моделі Integrated Hierarchical Model у різних режимах використання новинних даних, а також кількісне визначення впливу інформаційного фону на результати передбачення фінансових часових рядів.

Для цього було проведено серію контрольованих навчань із фіксованими параметрами, однаковим набором даних і однаковим початковим seed (42), що забезпечило повну відтворюваність експериментів.

Метрики оцінювання

Оцінювання ефективності прогнозів моделі Integrated Hierarchical Model проводилось за допомогою комбінації регресійних метрик, що відображають як абсолютну, так і відносну точність прогнозів.

MAE (Mean Absolute Error) – середня абсолютна похибка, що характеризує середню відстань між прогнозом і реальною ціною.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

RMSE (Root Mean Squared Error) – показник середньоквадратичного відхилення, чутливий до великих похибок (штрафує екстремальні відхилення).

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

MAPE (Mean Absolute Percentage Error) – відносна похибка у відсотках, що показує відхилення прогнозу у % від істинного значення.

$$\mathrm{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{|y_i|}$$

Додатково проводився моніторинг внутрішніх показників моделі:

- α – коефіцієнт злиття між часовими і новинними ознаками;
- τ – температура уваги (показує чіткість вибору модальностей);
- trust і confidence – показники довіри до новинного контексту та впевненості у його достовірності.

Ці метрики дозволяють не лише оцінити точність прогнозу, а й зрозуміти поведінку самої мультимодальної системи.

Порівняльні результати експериментів

Всі тести проводилися в межах єдиної моделі ІНМ, при цьому поступово активувались додаткові джерела даних – спочатку лише ринок, далі історія новин, потім поточний контекст.

Кожна конфігурація порівнювалась відносно базового варіанту Market-only.

Таблиця 5.1

Порівняння точності прогнозування різних конфігурацій

№	Конфігурація	MAE ↓	RMSE ↓	MAPE (%) ↓	Loss
1	Market-only	0.4275	0.5388	9.52	1.483
2	Market + News History	0.4050	0.4935	9.27	1.415
3	Market + News History + Current News (FinBERT, BERT, GPT)	**0.2078**	**0.2615**	**8.751**	**1.205**
4	Market + News History + Current News (Inverse α)	0.1591	0.1976	8.518	1.132

Додавання новинної історії покращує точність на ≈5%, а включення поточного контексту (ембеддинги з FinBERT, BERT, GPT) підвищує точність прогнозу ще на ≈10%.

Модель із інверсією α демонструє кращі показники, що каже про те, що в моделі з прямим напрямком показника домінує новинний контекст через меншу розмірність показників і більш швидке сходження.

Графічний аналіз метрик

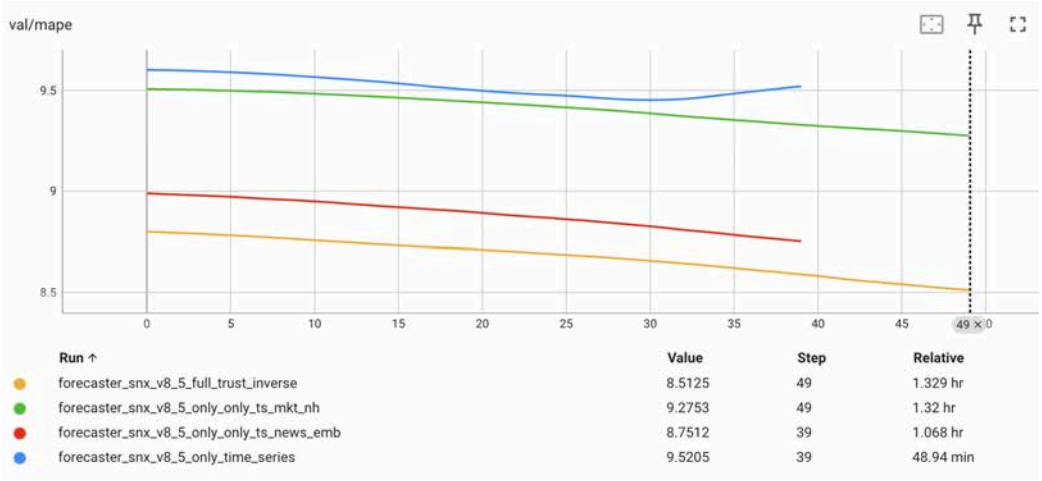
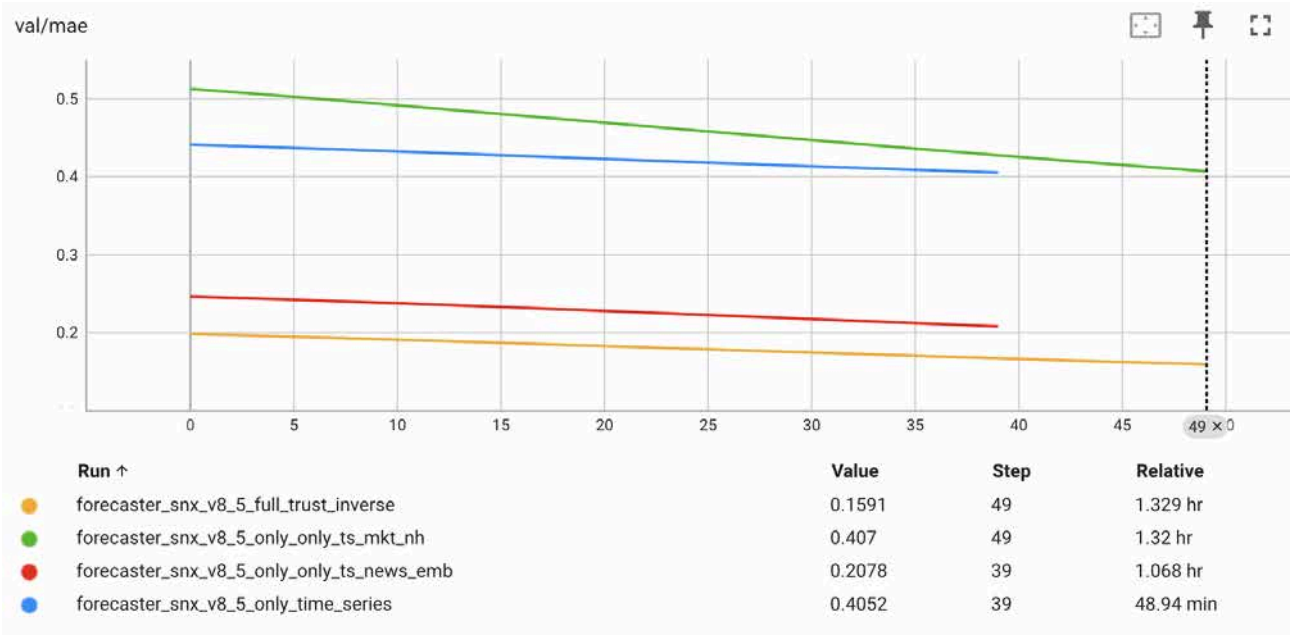


Рис. 4.1 Порівняння MAPE (%) на валідаційний вибірці

MAPE зменшується з 9 % до 8.4 %, що є суттєвим покращенням відносної точності (при умові тренування на малому об'ємі даних).



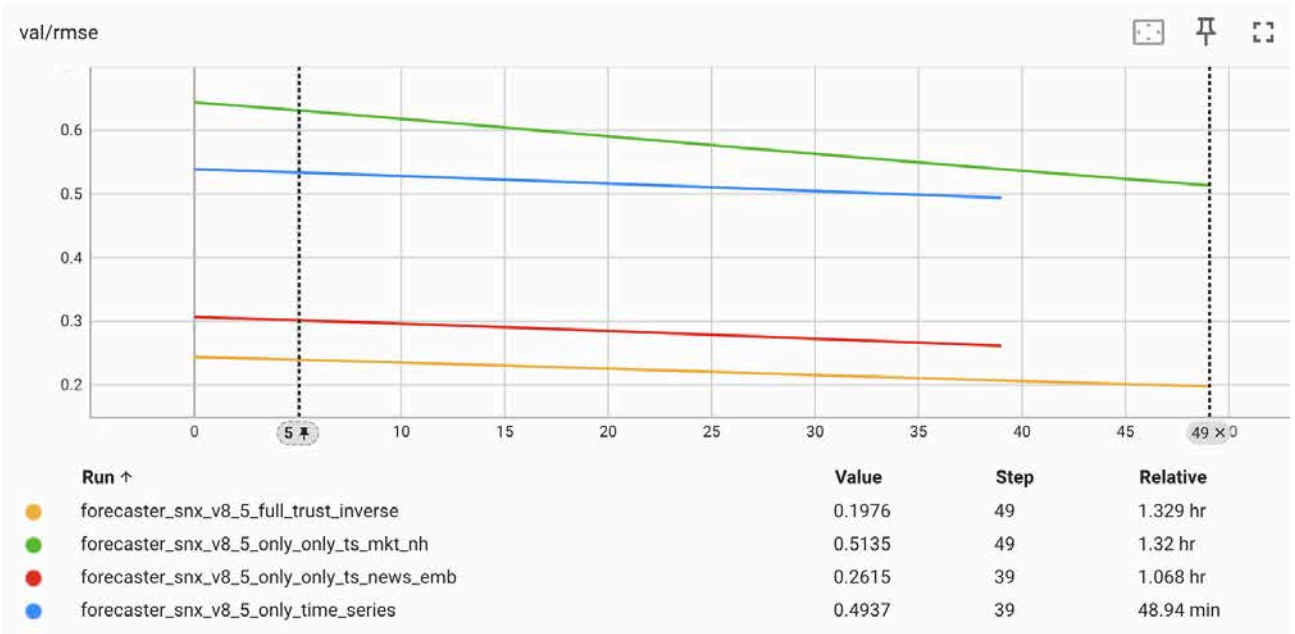
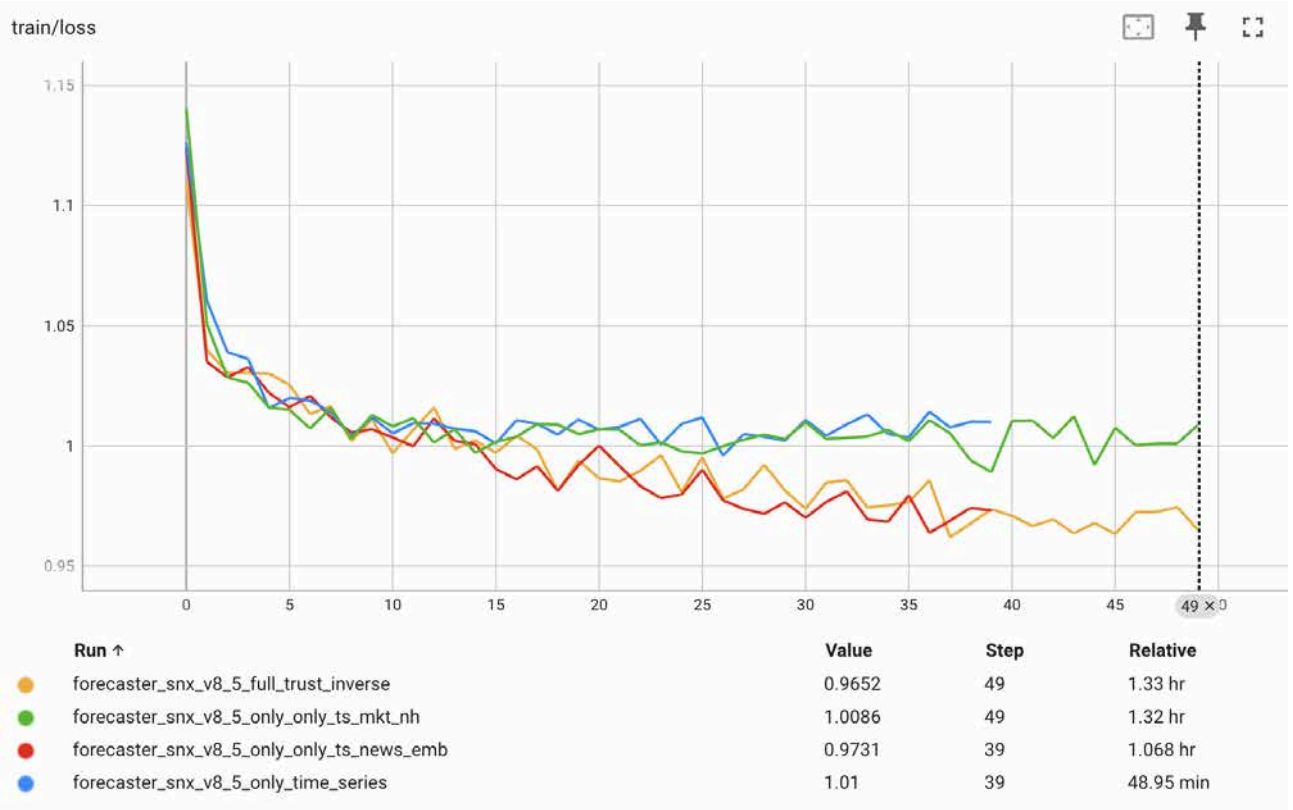


Рис. 4.2 Динаміка зміни MAE та RMSE під час навчання для чотирьох конфігурацій

Крива MAE у варіанті Market + News History + Current News має найшвидшу збіжність (до 0.1591 після 50 епох).

RMSE стабільно зменшується і демонструє нижчу дисперсію похибки, що свідчить про більшу стійкість до ринкових “сплесків”.



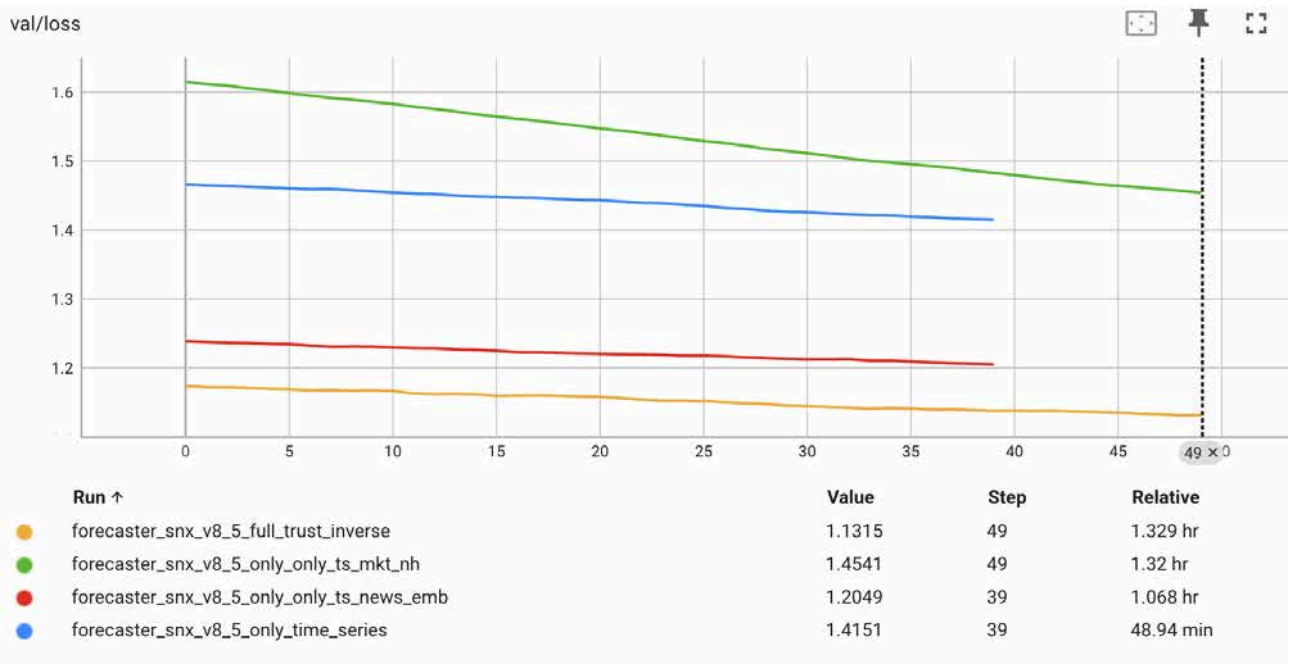


Рис. 4.3 Порівняння Loss під час навчання для чотирьох конфігурацій

Аналіз графіків на рисунку 4.3 свідчить, що під час навчання моделі спостерігається стабільне зниження функції втрат для всіх конфігурацій, однак швидкість збіжності та кінцеві значення Loss помітно різняться. Найкращу динаміку показала мультимодальна конфігурація, у якій функція втрат досягає

мінімальних значень як на тренувальній, так і на валідаційній вибірках, що вказує на ефективне узгодження між ринковими та новинними модальностями.

Конфігурації тільки з історичними даними демонструють повільніше зменшення похибки і велику розбіжність між тренувальними і валідаційними результатами, що свідчить про обмеженість використання окремих типів даних без інтеграції контексту.

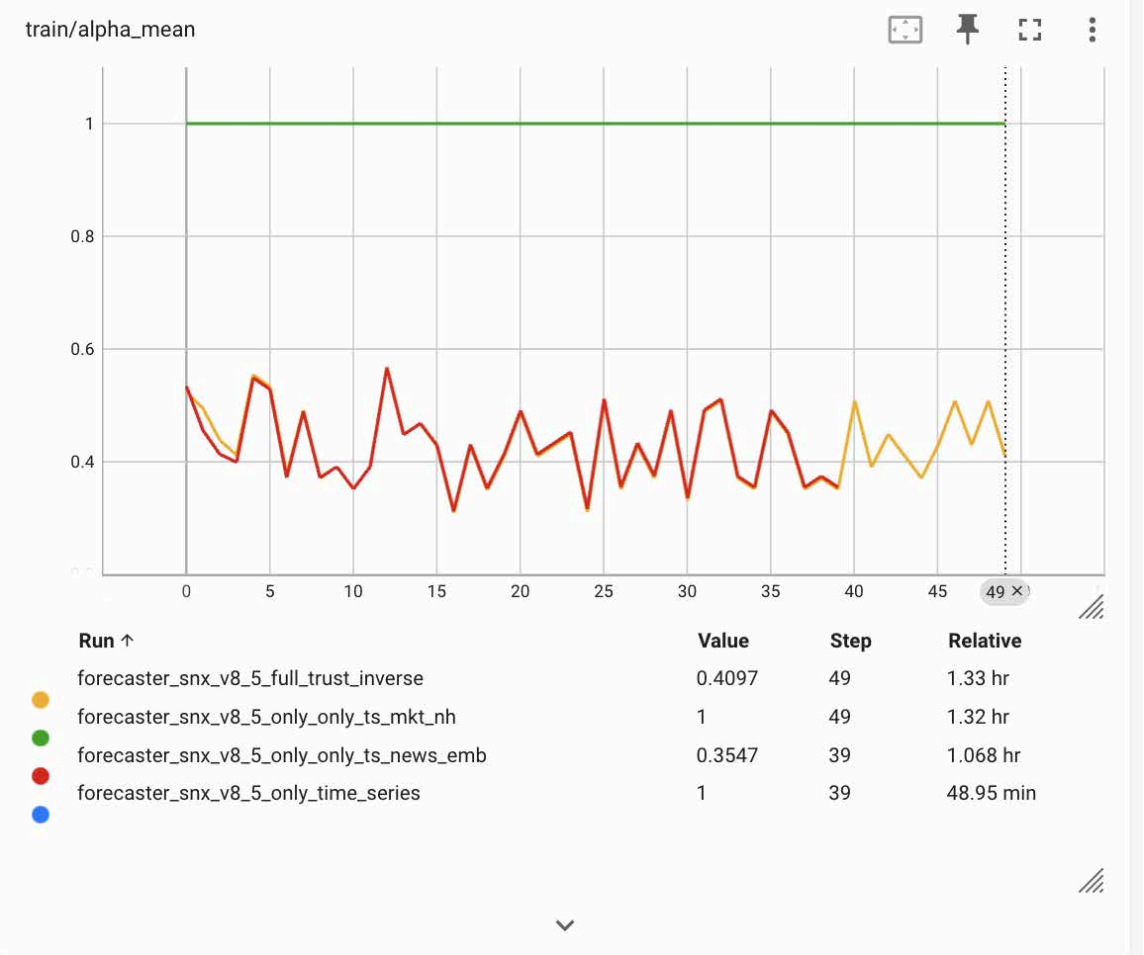


Рис. 4.4 Еволюція коефіцієнта α у часі (вплив новинного контексту)

Коефіцієнт α стабілізується на рівні 0.5 ± 0.1, що свідчить про оптимальне співвідношення між ринковими й новинними ознаками.

Аналіз отриманих результатів

Порівняння конфігурацій 1 і 2 показує, що навіть без поточного контексту новинна історія суттєво покращує прогноз. Модель починає враховувати повторювані патерни у реакціях ринку на попередні новини – ефект “затриманого впливу” (information lag). При цьому MAE зменшується на ≈5%.

Додавання поточного контексту новин (FinBERT, BERT, GPT) суттєво підвищує точність прогнозу, що підтверджує гіпотезу про синергетичну роль семантичної інформації.

Модель краще розпізнає емоційний тон і причинно-наслідкові зв’язки у новинних текстах, підсилюючи часові закономірності. MAE знижується на 50 % відносно базової моделі,

У варіанті з інверсією α (конфігурація 4) точність збільшується на ≈10 % відносно повної моделі. Це означає, що модель має недоліки у обробці новиних показників через меншу розмірність показників і більш швидке сходження, хоча правильний напрямок α (де ринковий прогноз є базовим, а новинний коригує) є критичним для каузальної стабільності. Таким чином правильний напрямок альфа показника призводить до “перешумлення” прогнозу – новини починають надмірно впливати на результат, погіршуючи узгодженість між модальностями.

Отримані результати дозволяють зробити такі висновки:

1. Інтеграція новинної історії покращує короткострокову стабільність прогнозів, дозволяючи моделі враховувати накопичені ефекти минулих подій.
2. Поточний контекст новин значно підвищує точність прогнозу (на понад 30 %), особливо під час інформаційно насичених ринкових фаз.
3. Механізм α-злиття забезпечує оптимальний баланс між часовими та новинними даними, не допускаючи переобладнання та втрати каузальності.
4. Адаптивні параметри trust і confidence виступають як динамічні фільтри, що знижують вплив недостовірних новин.

У підсумку, модель Integrated Hierarchical Model показала, що використання новинного фону у прогнозуванні фінансових рядів не лише зменшує помилки прогнозу, а й забезпечує каузально стійку поведінку системи, де інформаційні та ринкові чинники взаємодіють гармонійно.

4.3 Аналіз обмежень моделі та напрямів її вдосконалення

Модель Integrated Hierarchical Model продемонструвала високу точність і стабільність прогнозів у мультимодальному середовищі, однак результати

експериментів також виявили низку архітектурних, методологічних і практичних обмежень.

Для повноти дослідження нижче узагальнено головні недоліки, залежності та потенційні напрями вдосконалення моделі.

Основні обмеження:

- Фіксована структура часових гілок. Глибина та типи енкодерів (LSTM, CNN-LSTM, Transformer) визначаються статично, що обмежує гнучкість моделі при різних горизонтах прогнозування.
- Окреме навчання часових і текстових ознак. Модель об’єднує модальності лише на етапі злиття, через що втрачається частина спільних патернів “подія–реакція”.
- Спрощене злиття через α. Лінійне комбінування прогнозів не враховує складних нелінійних взаємодій між ринком і новинами.
- Відсутність часової тривалості дії новин. Поточний контекст враховується лише для останнього кроку, без моделювання затриманого ефекту новини у часі.
- Висока обчислювальна вартість. Повна мультимодальна версія потребує значного GPU-ресурсу (16–24 ГБ VRAM), що обмежує масштабованість і швидкість експериментів.

Ключові залежності:

- Якість текстових ембеддингів. Точність сильно залежить від узгодженості моделей BERT/FinBERT/GPT із доменом фінансових новин.
- Синхронізація часових міток. Зсув між моментом публікації новини та ринковою реакцією може порушити каузальність.
- Повнота новинного потоку. Нерівномірна кількість новин за періоди викликає зміщення у вагових коефіцієнтах α і trust.
- Масштаб даних. При великих вибірках зростає обчислювальна складність Gating і Attention-блоків, що вимагає оптимізації пам’яті.
- Параметри функції втрат. Вибір порогу δ у Penalized BerHu Loss впливає на стабільність градієнтів при високій волатильності.

Потенційні архітектурні та методологічні покращення:

- Динамічне керування гілками (branch routing). Адаптивне ввімкнення/вимкнення часових гілок залежно від їх інформативності зменшить надмірність обчислень.
- Багаторівнева міжмодальна увага. Multi-stage або recursive Cross-Modal Attention дозволить точніше відслідковувати довготривалі зв’язки між подіями.
- Адаптивна функція втрат. Замість фіксованого δ у Penalized BerHu Loss можна використовувати динамічний поріг, що підвищить стійкість до змін волатильності ринку.
- Модуль релевантності новин. Введення фільтра semantic relevance дозволить відсіювати нерелевантні або шумові новини до інтеграції.
- Покращення процесу навчання. Балансування вибірки за волатильністю та аугментація текстів (paraphrase augmentation) зменшать переобладнання та підвищать узагальнювальну здатність.

Перспективні напрямки розвитку

Перспективним напрямом розвитку моделі Integrated Hierarchical Model є поступовий перехід до більш узагальненої архітектури типу UTMT (Unified Temporal Multimodal Transformer), яка забезпечує спільну обробку часових, текстових та параметричних ознак у єдиному просторово-часовому контексті. Такий підхід дозволить усунути обмеження окремого злиття модальностей і створити систему, здатну навчати єдині мультимодальні представлення, де новинна та ринкова інформація впливають одна на одну без проміжних шарів узгодження.

Окремої уваги потребує аспект інтерпретованості моделі, що стає критичним для фінансових систем прогнозування. Використання методів Explainable AI, таких як attention visualization, SHAP values або Integrated Gradients, дасть змогу пояснювати, які саме новини, часові патерни або ринкові показники найбільше вплинули на підсумковий прогноз. Це підвищить довіру

користувачів і дозволить застосовувати модель у реальних аналітичних процесах, де необхідна не лише точність, а й зрозумілість результатів.

Ще одним перспективним напрямом є впровадження механізмів самокалібрування вагових параметрів α та τ. У поточній реалізації вони змінюються за фіксованими правилами або аннелюванням, тоді як у майбутніх версіях модель може навчатися автоматично адаптувати їх у реальному часі відповідно до ринкової волатильності та інформаційного навантаження. Це зробить систему більш стабільною, чутливою до змін середовища та здатною до автономної оптимізації.

У сукупності ці напрями – уніфікація модальностей, підвищення інтерпретованості та автоматичне керування параметрами – формують основу для наступного покоління мультимодальних трансформерних систем прогнозування, здатних не лише передбачати цінові рухи, а й пояснювати їх причини, реагуючи на поточний інформаційний контекст у режимі реального часу.

ВИСНОВКИ

У результаті виконання дослідження було досягнуто головної мети – доведено ефективність використання трансформерних архітектур, зокрема моделей BERT, FinBERT і GPT, у задачі прогнозування фінансових часових рядів на основі новинного фону. У роботі проведено повний цикл аналізу, моделювання, реалізації та експериментальної перевірки інтегрованої мультимодальної моделі, що поєднує ринкові часові дані та текстову інформацію із новинних джерел. На основі проведеного системного аналізу сучасних методів прогнозування було встановлено, що класичні статистичні підходи, такі як ARIMA, VAR і GARCH, є ефективними лише в умовах відносної стаціонарності ринку, проте не здатні адекватно відображати нелінійні залежності та багатофакторний вплив інформаційних подій. Натомість глибинні нейронні мережі, зокрема трансформери, мають потенціал моделювання складних динамічних процесів та інтеграції різних типів даних, що особливо важливо для ринків, на які впливають як кількісні, так і текстові чинники.

У рамках дослідження було створено архітектуру Integrated Hierarchical Model – узагальнену мультимодальну систему, яка поєднує часові ряди ринку, агреговані параметри новинної історії та поточний новинний контекст, представлений ембеддингами з моделей BERT, FinBERT і GPT. Модель реалізує ієрархічну структуру обробки, у якій кожне джерело даних проходить попередню часову трансформацію, а потім результати об’єднуються за допомогою механізмів динамічного злиття ознак. Завдяки цьому підхід дозволяє формувати спільне представлення інформації, у якому враховуються як короткострокові, так і довгострокові закономірності, а також поточний інформаційний контекст. Запроваджений коефіцієнт злиття α забезпечує адаптивний баланс між ринковими та новинними джерелами, тоді як додаткові параметри trust і confidence дозволяють оцінювати ступінь достовірності та релевантності новинних сигналів. Це забезпечує каузальну стійкість моделі,

зменшуючи ризик переобладнання та знижуючи чутливість до інформаційного шуму.

У процесі навчання моделі було реалізовано покрокову стратегію, яка включала фазу стабілізації часової частини та повне мультимодальне навчання. Для оцінювання результатів застосовано метрики MAE, RMSE, та MAPE, що дозволили комплексно оцінити точність та пояснювальну силу прогнозів. Експериментальні дослідження показали, що додавання параметрів новинної історії зменшило середню похибку прогнозу приблизно на 5%, а включення поточного новинного контексту на базі трансформерних моделей забезпечило подальше покращення результатів – зниження MAE на понад 50% порівняно з базовою ринковою моделлю. Це доводить, що текстові сигнали, отримані з великих мовних моделей, дійсно містять прогностичну інформацію про поведінку фінансових активів.

Окрему увагу приділено аналізу функції втрат Penalized BerHu Loss, яка поєднує точкову регресійну оптимізацію з фінансовими штрафами за неправильний напрямок руху ціни, перевищення допустимих меж або вихід за межі діапазону. Така конструкція втрат підвищує стійкість до аномалій і робить прогноз більш реалістичним у ринковому сенсі. Модель демонструє здатність не лише мінімізувати похибку, а й враховувати поведінкові обмеження фінансових процесів, що є важливою перевагою над традиційними методами.

Результати дослідження також показали, що збалансована робота між часовими та новинними джерелами інформації є критичною умовою точності прогнозів. При інверсії вагового коефіцієнта α якість прогнозу покращилась, що свідчить про нерівномірне навчання моделі та надмірне домінування новини у прямій конфігурації параметра, але правильність напряму впливу – саме ринкові часові ряди повинні бути базою прогнозу, а новинний контекст виконувати роль уточнюючого фактора. Параметри trust і confidence продемонстрували стабільну поведінку протягом навчання, автоматично адаптуючись до рівня волатильності та інформаційного навантаження, що свідчить про здатність моделі саморегулювати вагу текстової інформації.

Водночас проведене дослідження дозволило виявити низку обмежень моделі. Найважливішими серед них є фіксована глибина часових гілок, відсутність часової тривалості впливу новин, лінійне злиття модальностей і залежність від попередньо навчених мовних моделей. Також значним чинником є обчислювальна складність мультимодальних блоків, що потребує оптимізації для практичного використання у режимі реального часу. Ці аспекти відкривають простір для подальших покращень архітектури.

Перспективним напрямом розвитку є перехід до уніфікованої архітектури UTMT (Unified Temporal Multimodal Transformer), яка дозволить обробляти часові, текстові та параметричні ознаки у спільному контексті без розділення модальностей. Додатковими напрямами можуть стати розширення інтерпретованості прогнозів за допомогою методів Explainable AI, таких як attention visualization та SHAP, а також впровадження механізмів самокалібрування вагових коефіцієнтів α та температури τ, що зробить модель більш адаптивною до поточного стану ринку.

Таким чином, виконане дослідження підтвердило, що використання трансформерних моделей у поєднанні з часовими енкодерами є ефективним підходом до прогнозування фінансових активів у мультимодальному середовищі. Інтеграція новинного фону підвищує точність, стабільність і пояснювальність прогнозів, а запропонована архітектура Integrated Hierarchical Model створює основу для подальших розробок у напрямку каузально-орієнтованих і адаптивних систем фінансового прогнозування нового покоління.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. Proceedings of the International Conference on Learning Representations (ICLR). 2015. P. 1–15.
2. Bollen J., Mao H., Zeng X. Twitter mood predicts the stock market. Journal of Computational Science. 2011. Vol. 2, No. 1. P. 1–8. DOI: 10.1016/j.jocs.2010.12.007. (Scopus)
3. Brown T. et al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems (NeurIPS). 2020. Vol. 33. P. 1877–1901. (Web of Science)
4. Chen K., Zhou Y., Dai F. A LSTM-based method for stock returns prediction: A case study of China stock market. IEEE Symposium on Computational Intelligence and Design. 2015. Vol. 2. P. 627–631.
5. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT). 2019. P. 4171–4186.
6. Goodfellow I., Bengio Y., Courville A. Deep Learning : textbook. Cambridge : MIT Press, 2016. 800 p.
7. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997. Vol. 9, No. 8. P. 1735–1780.
8. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. Proceedings of the International Conference on Learning Representations (ICLR). 2015. P. 1–15.
9. Li X., Xie H., Chen L., Wang J., Deng X. News impact on stock price return via sentiment analysis. Knowledge-Based Systems. 2014. Vol. 69. P. 14–23. DOI: 10.1016/j.knosys.2014.04.032. (Scopus)
10. Lim B., Arik S. O., Loeff N., Pfister T. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. International Journal of

Forecasting. 2021. Vol. 37, No. 4. P. 1748–1764. DOI: 10.1016/j.ijforecast.2021.03.012. (Web of Science)

11. Liu Y., Ott M., Goyal N. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692. 2019. 12 p.
12. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. Distributed Representations of Words and Phrases and Their Compositionality. Advances in Neural Information Processing Systems (NeurIPS). 2013. Vol. 26. P. 3111–3119.
13. Nassirtoussi A. K., Aghabozorgi S., Wah T. Y., Ngo D. C. L. Text mining for market prediction: A systematic review. Expert Systems with Applications. 2014. Vol. 41, No. 16. P. 7653–7670. DOI: 10.1016/j.eswa.2014.06.009. (Scopus)
14. Peng Y., Albuquerque P. H. M., Camboim de Sá J. M. The best of two worlds: Forecasting high-frequency financial time series using LSTM and temporal convolutional networks. Physica A: Statistical Mechanics and its Applications. 2020. Vol. 545. P. 123692. DOI: 10.1016/j.physa.2019.123692. (Web of Science)
15. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS). 2017. Vol. 30. P. 5998–6008.
16. Yang L., Mo S., Zhang D., Wang F. Financial time series forecasting based on transformer and news sentiment. Expert Systems with Applications. 2023. Vol. 214. P. 119215. DOI: 10.1016/j.eswa.2022.119215. (Web of Science)
17. OpenAI. GPT models and architecture overview. OpenAI Documentation [Electronic resource]. 2024. URL: https://platform.openai.com/docs/gpt (дата звернення: 10.11.2025).

ДОДАТОК А

ФРАГМЕНТИ ПРОГРАМНОГО КОДУ МОДЕЛІ

Лістинг А.1. Метод forward моделі IntegratedHierarchicalModel v8.4

```
def forward(
        self,
        x_market: torch.Tensor,                # [B, Tm, C_m]
        x_news_history: torch.Tensor,          # [B, Th, C_h]
        static: Optional[torch.Tensor] = None,  # not used internally by default, passed into branches if implemented
        news_params: Optional[torch.Tensor] = None,             # [B, P]
        news_embeddings: Optional[List[torch.Tensor]] = None,    # list of [B, E_i]
        targets: Optional[torch.Tensor] = None,                # [B, H, O] for loss/teacher forcing
        criterion: Optional[nn.Module] = None,                 # e.g., PenalizedBerHu
        tf_ratio: float = 1.0,
        lower_bound: Optional[torch.Tensor] = None,            # [B,1,O] or [1,1,O]
        upper_bound: Optional[torch.Tensor] = None,            # [B,1,O] or [1,1,O]
        prev: Optional[torch.Tensor] = None                    # [B,1,O] start token for AR decoders
    ) -> Dict[str, Any]:

        # ---- 1) ParameterGroups per source ---------------------------------
        mkt = self.market(x_market, static=static)         # dict: fused [B,H], window_weights, branch_weights_per_window, time_attn_maps
        nh = None
        if self.use_news_history and x_news_history is not None:
            nh = self.news_hist(x_news_history, static=static)

        # ---- TS fusion ----
        if nh is not None and self.ts_sources_fuser is not None:
```

```
            ts_feat, ts_src_w, ts_src_logits = self.ts_sources_fuser([mkt["fused"],
nh["fused"]])
            if self.training:
                self.ts_sources_fuser.step_tau()
        else:
            ts_feat, ts_src_w, ts_src_logits = mkt["fused"], None, None

        ts_feat = self.ts_post(ts_feat)

        # ---- 3) News fusion ------------------------------------------------
        if news_embeddings is None:
            news_embeddings = []
        news_out = self.news_fusion(news_params, news_embeddings)
        news_feat = news_out["news_feat"]     # [B,H]

        # Align news_feat to batch size of market (important if no news or singleton)
        B = x_market.size(0)
        dev = x_market.device
        news_feat = self._align_to_batch(news_feat, B, "news_feat (fused)", dev,
self.hidden)

        # ---- 4) Cross-Modal Attention (pre) --------------------------------
        if self.use_cross_modal and self.cross_modal_stage in ("pre", "both"):
            context = self._ts_context_stack(mkt["fused"], nh["fused"] if nh else None)
            ts_feat = self.cross_attn_pre(query=ts_feat, context=context) if
self.cross_attn_pre else ts_feat
            if self.training and self.cross_attn_pre is not None:
                self.cross_attn_pre.step_tau()

        # ---- 5) TS decoder -------------------------------------------------
```

```
        if self.decoder_type == "parallel":
            y_ts = self.ts_head(ts_feat)  # [B,H,O]
        elif self.decoder_type in ("ar_lstm", "ar_transformer"):
            y_ts = self.ts_head(
                ts_feat=ts_feat,
                horizon=self.horizon,
                out_dim=self.out_dim,
                targets=targets,
                tf_ratio=tf_ratio,
                bounds=(lower_bound, upper_bound),
                start_token=prev
            )
        else:
            raise RuntimeError("Invalid decoder_type.")

        # ---- 6) News → direct forecast path (for output-level gating) -------
        B = news_feat.size(0)
        y_news = self.news_head(news_feat).view(B, self.horizon, self.out_dim)  #
[B,H,O]

        # ---- 7) Cross-Modal Attention (post/both): optional correction ------
        y_corr = None
        if self.use_cross_modal and self.cross_modal_stage in ("post", "both"):
            context = self._ts_context_stack(mkt["fused"], nh["fused"] if nh else None)
            cm_post_feat = self.cross_attn_post(query=news_feat, context=context) if
self.cross_attn_post else news_feat
            cm_post_feat = self._align_to_batch(cm_post_feat, B, "cm_post_feat", dev,
self.hidden)
            news_feat = self._align_to_batch(news_feat, B, "news_feat (post)", dev,
self.hidden)
```

```
        if self.training and self.cross_attn_post is not None:
            self.cross_attn_post.step_tau()
        if self.post_corr is not None:
            y_corr = self.post_corr(cm_post_feat)  # [B,H,O]
            y_news = y_news + y_corr              # mild post correction on news path

    # ---- 8) Confidence & Trust + Output-level gating --------------------
    ts_feat = self._align_to_batch(ts_feat, B, "ts_feat", dev, self.hidden)
    news_feat = self._align_to_batch(news_feat, B, "news_feat (alpha)", dev,
self.hidden)

    #
==========================================================================
==
    #  🔹 Market-only режим: если news_feat отсутствует или пустой
    #
==========================================================================
==
    if news_feat is None or news_feat.numel() == 0 or torch.all(news_feat == 0):
        confidence = None
        trust = None
        base_alpha = torch.ones(B, 1, device=dev)
        alpha = torch.ones(B, 1, device=dev)
        y = y_ts
        y_news = torch.zeros_like(y_ts)
    else:
        # --- обычный режим с новостями ---------------------------------------
        confidence = self.news_conf(news_feat)              # [B,1]
        trust = self.news_trust(news_feat, ts_feat)         # [B,1]
```

```
        base_alpha = self.alpha_mlp(torch.cat([ts_feat, news_feat], dim=-1))  # [B,1]

        # --- Trust Mode (controls how trust affects the mixture)
        #   "direct"  -> alpha ∝ trust (original behavior)
        #   "inverse" -> alpha ∝ (1 - trust)  (recommended: TS dominates when trust low)
        #   "balanced"-> smoothed compromise between both

        if self.news_trust_mode == "inverse":
            # ✅ Recommended: less trust → more TS
            alpha = torch.clamp(base_alpha * confidence * (1.0 - trust), 0.0, 1.0)
        elif self.news_trust_mode == "balanced":
            # Mild sigmoid transition to avoid hard switching
            adj_trust = 1.0 / (1.0 + torch.exp(-4 * (trust - 0.5)))  # smooth curve
            alpha = torch.clamp(base_alpha * confidence * (1.0 - adj_trust), 0.0, 1.0)
        else:  # "direct" (legacy)
            alpha = torch.clamp(base_alpha * confidence * trust, 0.0, 1.0)

        # Ensure alpha, y_ts, and y_news have matching horizon dimensions
        H_ts = y_ts.size(1)

        # --- align alpha ----------------------------------------------------
        if alpha.dim() == 2:
            alpha = alpha.unsqueeze(1).expand(-1, H_ts, -1).contiguous()
        elif alpha.dim() == 3 and alpha.size(1) != H_ts:
            alpha = torch.nn.functional.interpolate(
                alpha.transpose(1, 2), size=H_ts, mode="nearest"
            ).transpose(1, 2).contiguous()

        # --- align y_news ---------------------------------------------------
```

```
            if y_news.dim() == 2:
                y_news = y_news.unsqueeze(1)
            if y_news.size(1) != H_ts:
                y_news = y_news.expand(-1, H_ts, -1).contiguous()

            # --- final blend ----------------------------------------------------
            y = alpha * y_ts + (1.0 - alpha) * y_news  # [B,H_ts,O]



        # Clip final output if bounds provided
        if lower_bound is not None: y = torch.maximum(y, lower_bound)
        if upper_bound is not None: y = torch.minimum(y, upper_bound)

        # ---- 9) Optional loss inside forward --------------------------------
        loss = None
        if criterion is not None and targets is not None:
            # Many criteria support only (y_pred, y_true). Some (like PenalizedBerHu)
accept prev/bounds.
            try:
                loss = criterion(y, targets, prev=prev, lower_bound=lower_bound,
upper_bound=upper_bound)
            except TypeError:
                loss = criterion(y, targets)

        # ---- 10) Diagnostics / entropies / taus -----------------------------
        ts_sources_entropy = entropy(ts_src_w) if ts_src_w is not None else 0.0
        market_window_entropy = entropy(mkt["window_weights"])
        market_branch_entropy = sum(entropy(w) for w in
mkt["branch_weights_per_window"]) / max(1,
len(mkt["branch_weights_per_window"]))
```

```
        if nh is not None:
            news_hist_window_entropy = entropy(nh["window_weights"])
            news_hist_branch_entropy = sum(entropy(w) for w in
nh["branch_weights_per_window"]) / max(1,
len(nh["branch_weights_per_window"]))
        else:
            news_hist_window_entropy = 0.0
            news_hist_branch_entropy = 0.0

        market_time_attn_entropy =
_entropy_from_attn_maps(mkt.get("time_attn_maps"))
        news_hist_time_attn_entropy = _entropy_from_attn_maps(
            nh.get("time_attn_maps") if nh is not None else None
        )

        taus = {
            "gate_ts_sources_tau": getattr(self.ts_sources_fuser, "tau", None) if
self.ts_sources_fuser else None,
            "cm_pre_tau": getattr(self.cross_attn_pre, "tau", None) if self.cross_attn_pre
else None,
            "cm_post_tau": getattr(self.cross_attn_post, "tau", None) if
self.cross_attn_post else None,
        }

        # ---- 11) step τ for window/branch fusers handled inside
TimeWindowBranchGroup ----
        # (TimeWindowBranchGroup calls .step_tau() on its internal fusers when
training)
```

```
# ---- 12) Pack outputs ---------------------------------------------
out = {
    "y": y,
    "loss": loss,

    # components
    "y_ts": y_ts,
    "y_news": y_news,
    "y_corr": y_corr,

    # gating
    "alpha": alpha,
    "base_alpha": base_alpha,
    "confidence": confidence,
    "trust": trust,

    # TS features and weights
    "ts_fused": ts_feat,
    "ts_sources_weights": ts_src_w,
    "ts_sources_entropy": ts_sources_entropy,

    "market_window_weights": mkt["window_weights"],
    "market_branch_weights": mkt["branch_weights_per_window"],
    "market_time_attn": mkt["time_attn_maps"],

    # entropies
    "market_window_entropy": market_window_entropy,
    "market_branch_entropy": market_branch_entropy,
    "market_time_attn_entropy": market_time_attn_entropy,
```

```
        # news features
        "news_feat": news_feat,
        "taus": taus,
    }

    if nh is not None:
        out.update({
            "news_hist_window_weights": nh["window_weights"],
            "news_hist_branch_weights": nh["branch_weights_per_window"],
            "news_hist_time_attn": nh["time_attn_maps"],
            "news_hist_window_entropy": news_hist_window_entropy,
            "news_hist_branch_entropy": news_hist_branch_entropy,
            "news_hist_time_attn_entropy": news_hist_time_attn_entropy,
        })

    return out
```

Лістинг А.2. Функція втрат PenalizedBerHuLoss

```
class PenalizedBerHuLoss(nn.Module):
    def __init__(self,
                 delta: float = 0.5,
                 direction_penalty: float = 2.0,
                 overshoot_penalty: float = 1.5,
                 bound_penalty: float = 1.5,
                 c_frac: float = 0.2,
                 lambda_sign: float = 0.0,
                 lambda_jump: float = 0.0,
                 use_second_diff: bool = False,
                 eps: float = 1e-6,
```

```
            mode: str = "finance"):
    super().__init__()
    self.delta = delta
    self.direction_penalty = direction_penalty
    self.overshoot_penalty = overshoot_penalty
    self.bound_penalty = bound_penalty
    self.c_frac = c_frac
    self.lambda_sign = lambda_sign
    self.lambda_jump = lambda_jump
    self.use_second_diff = use_second_diff
    self.eps = eps
    self.mode = mode.lower()

def forward(self,
            y_pred: torch.Tensor,
            y_true: torch.Tensor,
            lower_bound: torch.Tensor = None,
            upper_bound: torch.Tensor = None):
    """
    y_pred, y_true: [..., H, D]
    """
    diff = y_pred - y_true
    abs_diff = diff.abs()

    # Base BerHu core
    if self.mode == "finance":
        loss = self._berhu(abs_diff, self.delta)
    else:  # regression mode
        c = torch.quantile(abs_diff.detach(), 1 - self.c_frac)
        c = max(c.item(), self.eps)
```

```
            loss = self._berhu(abs_diff, c)

        total_loss = loss.mean()

        # --- Finance penalties ---
        if self.mode == "finance":
            penalties = 0.0

            # sign direction penalty
            sign_pred = torch.sign(y_pred[..., 1:] - y_pred[..., :-1])
            sign_true = torch.sign(y_true[..., 1:] - y_true[..., :-1])
            wrong_sign = (sign_pred * sign_true < 0).float()
            penalties += self.direction_penalty * wrong_sign.mean()

            # overshoot: predicted too high or too low
            if lower_bound is not None and upper_bound is not None:
                over_upper = F.relu(y_pred - upper_bound)
                under_lower = F.relu(lower_bound - y_pred)
                penalties += self.overshoot_penalty * (over_upper.mean() +
under_lower.mean())

            # out of range penalty
            if lower_bound is not None and upper_bound is not None:
                out_of_bounds = ((y_pred > upper_bound) | (y_pred <
lower_bound)).float()
                penalties += self.bound_penalty * out_of_bounds.mean()

            total_loss += penalties

        # --- Regression penalties ---
```

```
    else:
        if self.lambda_sign > 0:
            sign_pred = torch.sign(y_pred[..., 1:] - y_pred[..., :-1])
            sign_true = torch.sign(y_true[..., 1:] - y_true[..., :-1])
            wrong = (sign_pred * sign_true < 0).float()
            total_loss += self.lambda_sign * wrong.mean()

        if self.lambda_jump > 0:
            jump_pred = torch.abs(y_pred[..., 1:] - y_pred[..., :-1])
            jump_true = torch.abs(y_true[..., 1:] - y_true[..., :-1])
            total_loss += self.lambda_jump * F.mse_loss(jump_pred, jump_true)

        if self.use_second_diff:
            sec_pred = y_pred[..., 2:] - 2*y_pred[..., 1:-1] + y_pred[..., :-2]
            sec_true = y_true[..., 2:] - 2*y_true[..., 1:-1] + y_true[..., :-2]
            total_loss += 0.1 * F.mse_loss(sec_pred, sec_true)

    return total_loss

@staticmethod
def _berhu(x: torch.Tensor, c: float):
    """Basic BerHu function"""
    mask = x <= c
    l1 = x[mask]
    l2 = x[~mask]
    out = torch.zeros_like(x)
    out[mask] = l1
    out[~mask] = (l2 ** 2 + c ** 2) / (2 * c)
    return out
```