Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**ПЕРЕДБАЧЕННЯ ЧАСОВИХ РЯДІВ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ**»

Виконав: студент 4-го року навчання,

Освітньої програми «Прикладна математика», 113

Тхорук Ігор Сергійович

Керівник Швай Н. О.___
кандидат фіз.-мат. наук, ст. викладач

Рецензент
_____
(прізвище та ініціали)

Кваліфікаційна робота захищена

з оцінкою
_____

Секретар ЕК
_____

«____» _____
20____ р.

Київ 2023

Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

ЗАТВЕРДЖУЮ
Зав.кафедри математики,
проф., доктор фіз.-мат. наук
_____*Олійник Б.В.*
(підпис)
“____”_____ 2023

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
для кваліфікаційної роботи
студенту 4-го курсу, факультету інформатики Тхоруку
Ігорю Сергійовичу

Тема: «Передбачення часових рядів за допомогою нейронних мереж»
Зміст кваліфікаційної роботи:

Анотація

1. Вступ
2. Передбачення часових рядів
3. Опис експериментів
4. Результати

Висновки

Список літератури

Дата видачі “ ____”_____ 2023 Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Зміст

Зміст……3

Анотація……4

1 Вступ……5

1.1 Актуальність……..5

1.2 Мета, завдання дослідження……...6

2 Передбачення часових рядів……....7

2.1 Постановка задачі……7

2.2 Базовий метод……8

2.3 Повнозв'язний та згортковий шари……..8

2.3.1 Повнозв'язний шар……8

2.3.2 Згортковий шар……..9

2.4 Рекурентні нейронні мережі……..11

2.4.1 GRU……12

2.4.2 LSTM……...13

2.5 Методи оптимізації……..15

3 Опис експериментів……17

3.1 Дані……17

3.2 Моделі……17

3.3 Тренування……...18

4 Результати……..20

Висновки……...23

Список літератури……...24

Анотація

Метою цієї кваліфікаційної роботи є дослідження архітектур нейронних мереж для передбачення часових рядів. У роботі розглядались архітектури побудовані з повнозв'язних, згорткових та рекурентних шарів. Отримано результати їх застосування прямим та ітеративним методами, була проведена загальна та порівняльна характеристика моделей.

1 Вступ

1.1 Актуальність

У сучасному світі, ми стикаємося з величезним обсягом даних, що постійно генеруються і зберігаються. Ця масивна кількість інформації надає нам потенціал для отримання цінної інформації та прогнозування подій у майбутньому. Одним із ключових аспектів цього є передбачення часових рядів.

Часові ряди – це послідовності даних, де кожне спостереження має відповідне положення в часі. Прикладами часових рядів можуть бути щоденні фінансові показники, дані про температуру, заповненість доріг, споживання електроенергії тощо.

Передбачення часових рядів має велике значення в різних сферах. Наприклад, у фінансовому секторі, аналітики можуть використовувати дані про попередні показники фінансового ринку для прогнозування майбутніх трендів і прийняття стратегічних рішень щодо інвестування. У медицині, аналіз часових рядів може допомогти передбачити розвиток захворювань, контролювати популяційні зміни і вживати необхідні заходи безпеки. У сфері транспорту, прогнозування часових рядів може сприяти в управлінні трафіком, розкладу транспорту та оптимізації маршрутів.

Останні досягнення в області обробки природної мови, зокрема використання трансформерів [1], що є потужними моделями для розуміння та генерації тексту, привернули значну увагу до їх потенційного застосування в передбаченні часових рядів. Трансформери виявилися дуже успішними в багатьох завданнях обробки природної мови, включаючи машинний переклад, генерацію тексту та розпізнавання мови.

Однак, коли ми розглядаємо їх застосування до передбачення часових рядів, де метою є прогнозування майбутніх значень на основі попередніх даних, дослідники виявили, що точність трансформерів у цій задачі виявляється досить

низькою порівняно зі звичайними лінійними моделями. Це було підтверджено у дослідженні, опублікованому у статті "Are Transformers Effective for Time Series Forecasting?" [2], де автори порівняли результати передбачення часових рядів, отримані за допомогою трансформерів та звичайних лінійних моделей. Виявилося, що трансформери демонструють набагато нижчу точність і меншу здатність моделювати складні шаблони часових рядів порівняно зі звичайними лінійними моделями.

Тому в цій кваліфікаційній роботі було досліджено ефективність різних класичних нейронних мереж для передбачення часових рядів. Було перевірено різні архітектури моделей, та визначено, яка з них є найкращою та як вона впливає на точність передбачення.

1.2 Мета, завдання дослідження

Метою кваліфікаційної роботи є дослідження різних архітектур нейронних мереж для передбачення часових рядів, щоб порівняти їх точність передбачення.

В даному дослідженні були виконанні такі завдання:

- Імплементація моделей для передбачення часових рядів з використанням класичних архітектур шарів нейронних мереж
- Проведення експериментів на трьох різних датасетах
- Аналіз отриманих результатів

2 Передбачення часових рядів

2.1 Постановка задачі

Нехай часовий ряд складається з C змінних, та маємо історичні данні $X = \{X_1^t, X_2^t, \dots, X_C^t\}_{t=1}^{L}$, де L – довжина ряду в часі та X_i^t – значення i-тої змінної в час t. Завдання передбачення часового ряду полягає в передбачені значеннь $\widehat{X} = \{\widehat{X_1^t}, \widehat{X_2^t}, \dots \widehat{X_C^t}\}_{t=L+1}^{L+T}$ для T – майбутніх кроків.

Коли $T > 0$, задачу можна розв'язувати двома способами - пряме передбачення (direct multi-step forecasting, DMS)[6] або ітеративне передбачення (iterative multi-step forecasting, IMS)[5].

При використанні DMS передбачається прямо T кроків вперед. Передбачення за допомогою прямого методу краще використовувати, якщо T велике або якщо тяжко отримати точну модель для передбачення одного короку.

При використанні IMS ми передбачаємо один майбутній крок і використовуєм його як вхід для передбачення наступного. Ітеративний метод краще використовувати, якщо T малий і є точна модель для передбачення одного кроку.

Для оцінки точності передбачення використовується середня квадратична помилка (Mean square error або MSE):

$$L(\hat{X}, Y) = \frac{1}{TC} \sum_{i=0}^{C} \sum_{t=L}^{L+T} \left(\widehat{X_i^t} - Y_i^t\right)^2$$

де $\hat{X}$ – реальні значення часового ряду, Y – передбаченні значення часового ряду.

2.2 Базовий метод

Оскільки досліджуватимуться класичні архітектури, базовим методом для цього дослідження буде наївний метод або рухоме середнє (що має кращу точність), де передбачення на кожному майбутньому часовому кроці t зводиться до повторення останнього відомого значення або середнього останіх L кроків.

Цей метод є дуже простим і його ефективність може бути обмеженою, особливо якщо в часовому ряді є складні залежності та коливання (Рис. 1). Проте він все ще має хорошу точність порівняно з іншими більш складними методами на неперіодичних випадкових датасетах або при малому T. [7]

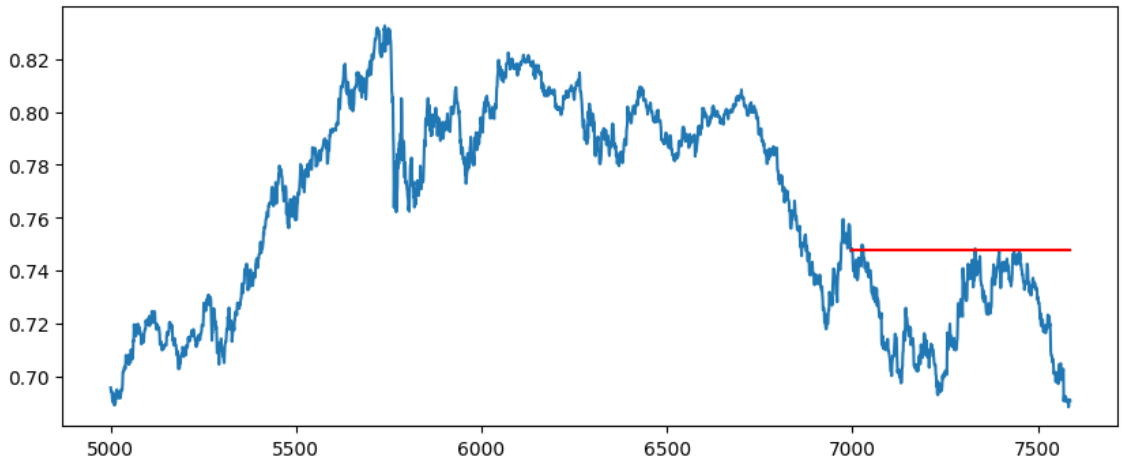

Рис.1 Зображення наївного методу. Синім позначено реальні данні, червоним передбачення.

2.3 Повнозв'язний та згортковий шари

2.3.1 Повнозв'язний шар

Повнозв'язна нейронна мережа, також відома як персептрон, є однією з найпростіших та найпоширеніших архітектур нейронних мереж. Вона

складається з вхідного шару, декількох прихованих шарів та вихідного шару (Рис. 2).

Формально повнозв'язний шар визначається:

$$X_n = \sigma(W_n X_{n-1} + B_n),$$

де X_n – вектор отриманий при застосуванні n-го шару, X_{n-1} – вектор виходів минулого шару, W_n, B_n – параметри, а σ – деяка не лінійна функція активації.

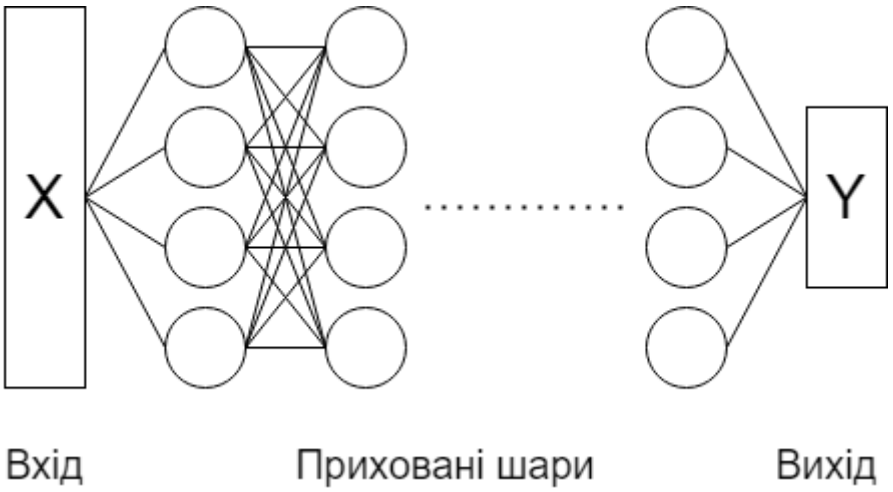

Рис.2 Зображення повнозв'язної нейронної мережі.

Для задачі передбачення часового ряду прямим передбаченням повонозв'язну нейронну мережу можна застосувати задавши вхід як вектор з елементів множини X, а вихід як вектор з елементів множини $\hat{X}$.

Для ітеративного передбачення ми так само задаємо вхід як вектор з елементів X, а вихід як вектор з елементів єдиного майбутнього кроку $\widehat{X^{L+1}}$. Далі задаємо вхід як X = $\{X_1^t, X_2^t, \dots, X_i^t\}_{t=2}^{L+1}$, де $X^{L+1} = \widehat{X^{L+1}}$ та ітеративно повторюємо це поки не отримаємо T майбутніх передбачень.

2.3.2 Згортковий шар

Основна ідея згорткової мережі полягає у тому, що вона використовує концепцію фільтрів, які шукають певні шаблони у вхідних даних. Фільтри є

невеликими матрицями ваг, які здійснюють операцію згортки з вхідними даними. Ця операція полягає в тому, що фільтр переміщується по всіх можливих позиціях вхідних даних і здійснює множення елементів фільтра на елементи вхідних даних, після чого результати додаються (Рис. 3).

Формально, 1D згорткова операція використовується для обробки одновимірних даних, таких як часові ряди. Нехай маємо вхідний одновимірний сигнал X_i довжиною L, та фільтр (ядро) h довжиною M. Згорткова операція між X та h обчислюється наступним чином:

$$Y_i^t = (X_i * h)^t = \sum_{k=0}^{M-1} X_i^{t+k-\left\lfloor \frac{k}{2} \right\rfloor} h_i^k$$

де Y_i – вихідний сигнал після конволюції, t – позиція у вихідному сигналі, $X_i^{t+k-\left\lfloor \frac{k}{2} \right\rfloor}$ – елемент i-тої вхідної змінної на позиції $t + k - \left\lfloor \frac{k}{2} \right\rfloor$, h_i^k – елемент i-го фільтру (ядра) на позиції k.

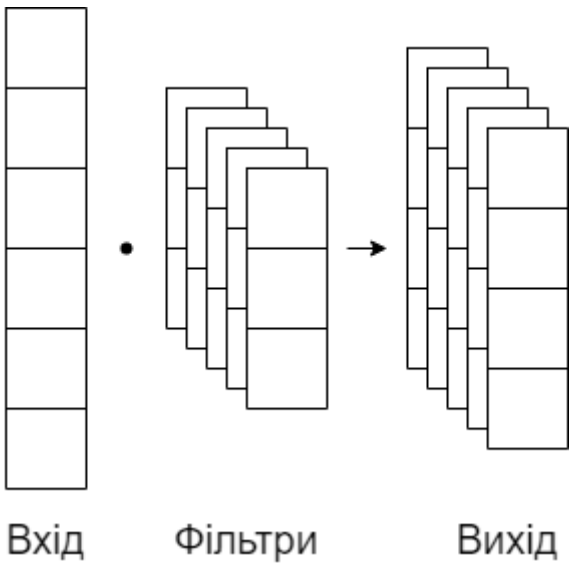

Рис.3 Зображення згорткового шару.

Звідси вихідний сигнал Y є результатом послідовного зсуву фільтра h по вхідному сигналу X та обчислення скалярного добутку між фільтром та

підмножиною вхідного сигналу на кожному кроці. Кожне значення Y^t відповідає вагованій сумі елементів вхідного сигналу залежно від положення фільтра.

Ця операція дозволяє виділити локальні шаблони або особливості в часовому ряді, допомагає виявляти залежності між значеннями сигналу та може бути використана для передбачення та аналізу часових рядів.

2.4 Рекурентні нейронні мережі

Рекурентні нейронні мережі (Recurrent Neural Networks або RNN) є класом нейронних мереж, які призначені для обробки та моделювання послідовних даних, таких як текст, звук, часові ряди тощо. Основна відмінність RNN від інших архітектур полягає в тому, що вони мають внутрішні стани (пам'ять), що дозволяє їм враховувати контекст з попередніх кроків, для обчислення цільової змінної на поточному кроці. Така особливість дозволяє обробляти вхідний ряд будь-якої довжини.

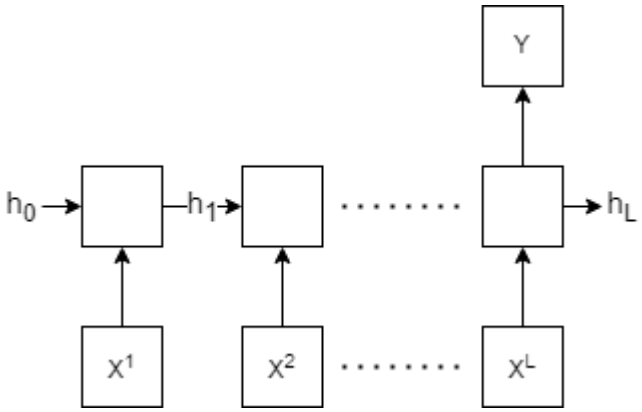

Рис.4 Зображення рекурентної нейронної мережі.

Формально рекурентний шар визначається:

$$H_t = \sigma(WX^t + UH_{t-1} + B),$$

де H_t - внутрішній стан, H_{t-1} - внутрішній стан минулого кроку, X^t - вхідне значення на t-ому кроці, W, U, B - параметри, σ - деяка нелінійна функція.

Щоб отримати цільову змінну $\hat{X}$ можна пропустити внутрішній стан останнього кроку через інший шар, наприклад повнозв'язний.

Однак звичайна рекурентна нейронна мережа страждає від проблеми затухання або вибуху градієнту [9], що негативно впливає на здатність моделі враховувати довгосторокові залежності. Тому в цій роботі були використанні такі покращення RNN як рекурентний блок з воротами (Gated Recurent Unit або GRU) та довгострокова короткострокова пам'ять (Long Short-Term Memory або LSTM).

2.4.1 GRU

GRU (Gated Recurent Unit) [4] є однією з варіацій рекурентних нейронних мереж, яка була розроблена для подолання проблеми зникнення градієнту. GRU використовує два ключових механізми, щоб зберегти та передавати важливу інформацію на тривалий період часу: ворота відкидання (reset gate) та ворота оновлення (update gate).

Ворота відкидання в GRU відповідають за регулювання того, яка інформація з попереднього кроку буде проігнорована. Це дає змогу моделі вирішувати, коли слід забути певну інформацію.

Ворота оновлення в GRU відповідає за вирішення, яка частина нової інформації буде врахована у внутрішньому стані моделі. Це дозволяє моделі вирішувати, коли слід оновити свій внутрішній стан залежно від нової інформації.

Ці два механізми допомагають контролювати потік інформації, дозволяючи моделі довше зберігати та використовувати важливі залежності. В результаті GRU може легше передавати градієнт під час тренування, зменшуючи проблему затухання градієнту. [8]

Формально GRU шар визначається:

$$Z^t = \sigma_g(W_z X^t + U_z H^{t-1} + B_z),$$

$$R^t = \sigma_g(W_r X^t + U_r H^{t-1} + B_r),$$

$$\widehat{H^t} = \sigma_h(W_h X^t + U_h(R^t \odot H^{t-1}) + B_h),$$

$$H^t = (1 - Z^t) \odot H^{t-1} + Z^t \odot \widehat{H^t},$$

де X^t – вхідний вектор, H^{t-1} – внутрішній стан минулого кроку, σ_g – сигмоїда, σ_h – функція tanh, Z^t – вектор воріт оновлення, R^t – вектор воріт відкидання, $\widehat{H^t}$ – вектор кандитат, H^t – внутрішній стан кроку або вихід, W, U, B – параметри.

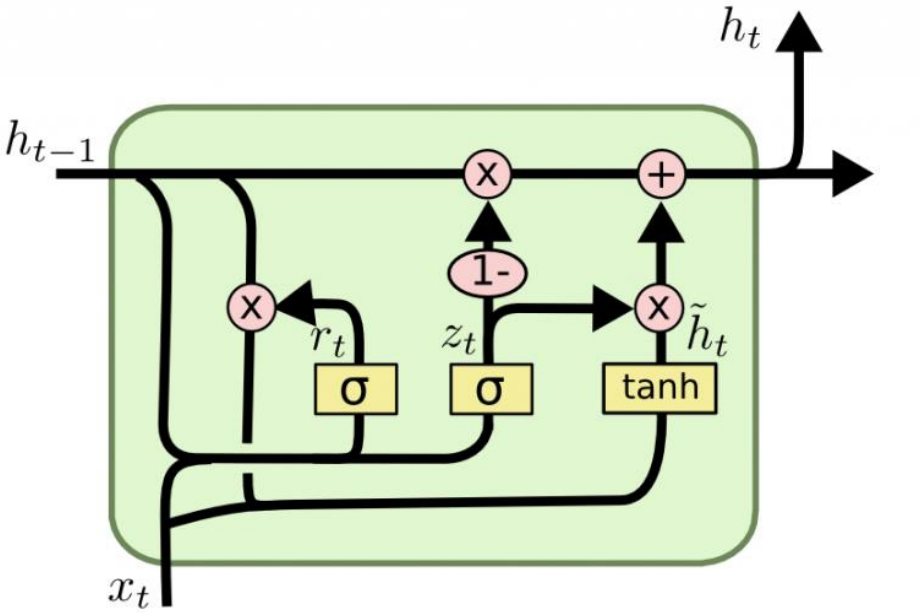

Рис.5 Зображення GRU блоку. [13]

2.4.2 LSTM

LSTM (Long Short-Term Memory) [3] є ще однією варіацією рекурентних нейронних мереж, спрямованою на подолання проблеми зникнення градієнту. LSTM використовує спеціальну архітектуру, що дозволяє моделі довше зберігати та використовувати важливу інформацію у тривалих залежностях.

Основна відмінність LSTM від стандартних рекурентних мереж полягає в наявності клітинного стану (cell state) та трьох воріт: ворота забування (forget gate), ворота входу (input gate), ворота виходу (output gate).

Ворота забуття визначають, яка інформація повинна бути забута у попередньому клітинному стані. Це дозволяє моделі вибирати, яку інформацію враховувати і яку проігнорувати.

Ворота входу визначають, яка нова інформація повинна бути додана до клітинного стану. Вони фільтрують вхід і визначають, яка інформація є важливою для оновлення стану.

Ворота виходу визначають, яка частина клітинного стану буде використовуватись для обчислення вихідного значення. Воно регулює, яка інформація з клітинного стану буде врахована у вихідному значенні.

Ці ворота контролюють потік інформації в LSTM шарі, дозволяючи моделі відділяти та зберігати важливі залежності на тривалий період часу. Клітинний стан допомагає моделі зберегти попередню інформацію та уникнути її надлишкового затухання або зростання. Це дозволяє LSTM передавати градієнт через тривалі залежності, запобігаючи проблемі затухання градієнту. [11]

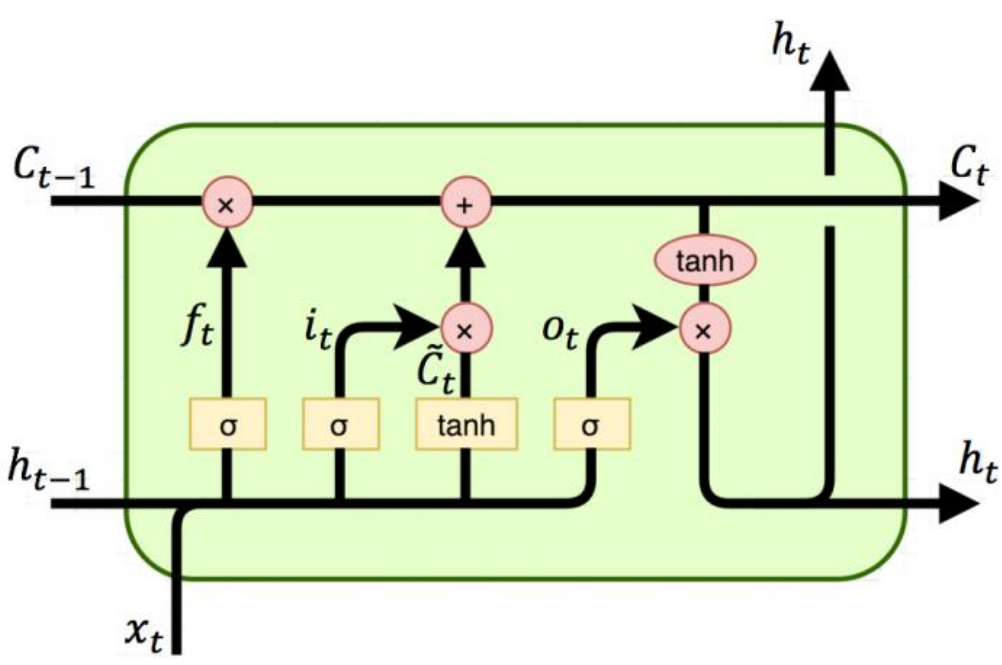

Рис.6 Зображення LSTM блоку. [13]

Формально LSTM шар визначається:

$$F^t = \sigma_g\left(W_f X^t + U_f H^{t-1} + B_f\right),$$

$$I^t = \sigma_g(W_i X^t + U_i H^{t-1} + B_i),$$

$$O\hat{}t \; = \; \sigma\_g(W_o X^t + U_o H^{t-1} + B_o,$$

$$\widehat{C^t} = \sigma_h(W_c V^t + U_c H^{t-1} + B_c),$$

$$C^t = F^t \odot C^{t-1} + I^t \odot \widehat{C^t},$$

$$H^t = O^t \odot \sigma_h(C^t),$$

де X^t – вхідний вектор, H^{t-1} – внутрішній стан минулого кроку, σ_g – сигмоїда, σ_h – функція tanh, F^t – вектор воріт забування, I^t – вектор воріт входу, O^t – вектор воріт виходу, $\widehat{C^t}$ – вектор кандидата клітинного стану, C^t – вектор клітинного стану, H^t – внутрішній стан кроку або вихід, W, U, B – параметри.

2.5 Метод оптимізації

Кожна модель представляє собою деяку функцію, що залежить від вхідних данних та параметрів. Параметри є змінними які підбираються за допомогою методу оптимізації, щоб вирішувати поставлену задачу. В цій роботі використовувався метод оптимізації - Adam (Adaptive Momentum Estimation) [14].

Adam є популярним методом оптимізації для тренування нейронних мереж, особливо в глибокому навчанні. Він комбінує в собі ідеї з методу моментів та адаптивного швидкісного градієнтного спуску, що дозволяє швидко досягти точки оптимуму.

Adam використовує експоненційно зважений середній квадрат градієнтів (Exponential Weighted Moving Average, EWMA) для оновлення параметрів моделі. Це забезпечує адаптивність швидкості навчання для кожного параметра окремо.

Формально Adam визначається:

$$g_s = \Delta_Q L(Q_{s-1}),$$

$$m_s = \beta_1 \cdot m_{s-1} + (1 - \beta_1) \cdot g_s,$$

$$v_s = \beta_2 \cdot v_{s-1} + (1 - \beta_2) \cdot g_s^2,$$

$$\widehat{m_s} = \frac{m_s}{(1 - \beta_1^s)},$$

$$\widehat{v_s} = \frac{v_s}{(1 - \beta_2^s)},$$

$$Q_s = Q_{s-1} - \alpha \cdot \frac{\widehat{m_s}}{\sqrt{\widehat{v_s}} + \epsilon},$$

де Q_s, Q_{s-1} – параметри моделі на тренувальному кроці s та $s-1$, $\Delta_Q L(Q_{s-1})$ - чаткова похідна функції втрат по параметрам Q в точці Q_{s-1}, α, β_1, β_2, ϵ - гіперпараметри.

3 Опис експериментів

3.1 Дані

Було взято 3 датасети з репозиторію "Multivariate Time series Data sets"[10]:

exchange rate. Колекція щоденних курсів валют восьми іноземних країн, включаючи Австралію, Великобританію, Канаду, Швейцарію, Китай, Японію, Нову Зеландію та Сінгапур, з 1990 по 2016 рік

solar AL. Датасет містить дані про виробництво сонячної енергії за 2006 рік, вибірка якої кожні 10 хвилин береться на 137 фотоелектричних установках штату Алабама.

traffic. Колекція щогодинних даних за 48 місяців (2015-2016) від Департаменту транспорту Каліфорнії. Ці дані описують рівень заповненості доріг (від 0 до 1), виміряний різними датчиками на шоссе у районі затоки Сан-Франциско. Для передбачення було взято 150 датчиків.

Всі датасети були нормалізовані, щоб середнє дорівнювало 0 а дисперсія була рівна 1.

3.2 Моделі

Для експерименту використовувались 6 архітектур нейронних мереж:

LinearNN. Модель складається лише з двох повнозв'язних шарів – вхідного та вихідного. Розмір вхідного шару дорівнює L а вихідного T

CNN LinearNN. Модель складається зі згорткового та повнозв'язного шарів. Згортковий шар складається з п’яти фільтрів довжини – три і до вхідних даних додається падінг, щоб вихідний ряд мав таку ж саму довжину. Далі до отриманого ряду застосовується ReLU та повнозв’язний шар з виходом розмірності T.

GRU. Модель складається з GRU та повнозв'зяного шарів. GRU шар має вхідний розмір – 1, та розмір внутрішнього шару L. Далі до виходу останього блоку застосовується повнозв’язний шар з виходом розмірності T.

CNN GRU. Модель складається зі згорткового, GRU та повнозв'язного шарів. Згортковий шар складається з п’яти фільтрів довжини – три і до вхідних даних додається падинг, щоб вихідний ряд мав таку ж саму довжину. Далі до отриманого ряду застосовується GRU шар, що має вхідний розмір – 1, та розмір внутрішнього шару L. Після цього до виходу останього блоку застосовується повнозв’язний шар з виходом розмірності T.

LSTM. Модель складається з LSTM та повнозв'язного шарів. LSTM шар має вхідний розмір – 1, та розмір внутрішнього шару L. Далі до виходу останього блоку застосовується повнозв’язний шар з виходом розмірноситі T.

CNN LSTM. Модель складається зі згорткового, LSTM та повнозв'язного шарів. Згортковий шар складається з п’яти фільтрів довжини – три і до вхідних даних додається падинг, щоб вихідний ряд мав таку ж саму довжину. Далі до отриманого ряду застосовується LSTM шар, що має вхідний розмір – 1, та розмір внутрішнього шару L. Після цього до виходу останього блоку застосовується повнозв’язний шар з виходом розмірності T.

Імплементовані моделі були на мові програмування python з використанням бібліотеки pytorch.

3.3 Тренування

Для тренування датасети розбивались на пари векторів X та $\hat{X}$ довжиною L та T. Далі отримані пари ділились на тренувальний, перевірочний та тестовий датасети за відношенням $70\%: 15\%: 15\%$.

Для кожного датасету та архітектури тренувалося 8 моделей з вхідними розмір L рівним 64 та 128 та вихідними T які дорівнюють 1, 4, 16, 64. Для моделей з рекурентними шарами замість вхідного розміру встановлювався розмір внутрішнього стану, оскільки вони не залежать від розміру вхідних даних.

Кожна модель тренувалась 20 епох і вибиралась модель епохи, де вона мала найменшу MSE на перевірочному датасеті. Після цього для обчислення кінцевої точності моделі використовувався тестові данні.

4 Результати

model	method	exchange rate				solar AL				traffic			
		1	4	16	64	1	4	16	64	1	4	16	64
LinearNN	DMS	0.009	0.011	0.017	0.044	0.012	0.035	0.118	0.217	0.242	0.357	0.415	0.419
	IMS		0.011	0.014	**0.022**		0.034	**0.082**	**0.143**		0.357	0.389	0.430
CNN LinearNN	DMS	0.019	0.011	0.024	0.101	0.008	0.031	0.108	0.189	0.191	0.309	0.387	0.403
	IMS		0.025	0.040	0.081		0.034	0.120	inf		0.342	5.589	1.382
GRU	DMS	0.006	0.009	0.018	0.064	**0.007**	**0.028**	0.126	0.402	0.181	0.311	0.397	0.523
	IMS		0.009	0.018	0.044		0.032	0.135	1.258		0.331	0.468	0.975
CNN GRU	DMS	0.012	0.012	0.032	0.075	**0.007**	0.031	0.140	0.384	**0.167**	**0.247**	**0.323**	**0.387**
	IMS		0.015	0.024	0.059		0.032	0.122	0.488		0.282	0.368	0.718
LSTM	DMS	0.008	0.012	0.021	0.077	**0.007**	**0.028**	0.141	0.396	0.169	0.284	0.359	0.461
	IMS		0.011	0.019	0.043		0.031	0.115	0.446		0.310	0.427	0.562
CNN LSTM	DMS	0.008	0.020	0.064	0.129	**0.007**	0.029	0.146	0.407	0.169	0.254	0.350	0.448
	IMS		0.014	0.031	0.131		0.041	0.129	0.420		0.277	0.367	0.782
Baseline	-	**0.003**	**0.005**	**0.012**	0.034	0.010	0.044	0.280	0.791	0.383	1.107	1.154	1.144

Табл. 1. MSE моделей на трьох датасетах з різною довжиною передбачення (1, 4, 16, 64). Жирним тестом позначенні найкращі точності в стовбці.

exchange rate. На цьому датасеті жодна з моделей не змогла перевершити базовий метод при передбачені на 1, 4 або 16 кроків вперед. Лише при передбаченні на 64 кроки вперед LinearNN показала кращу точність.

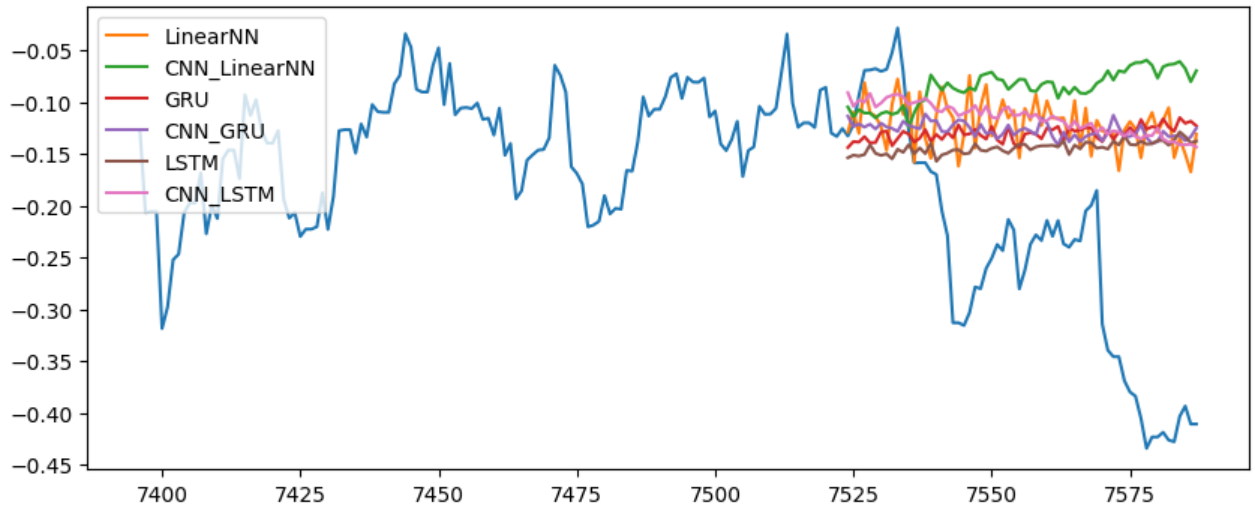

Рис.7 DMS передбачення на 64 кроки моделей датасету exchange rate

Курси валют часто найкраще апроксимуються випадковим блуканням (random walk) [12], тому не дивно, що в даному експерименті моделі не змогли перевершити базовий метод.

solar AL. На цьому датасеті моделі які використовували GRU та LSTM показали найкращу точність при передбаченні на малу кількість кроків (1, 4) в той час як LinearNN мала найкращу точність при передбаченні на довший проміжок часу (16, 64)

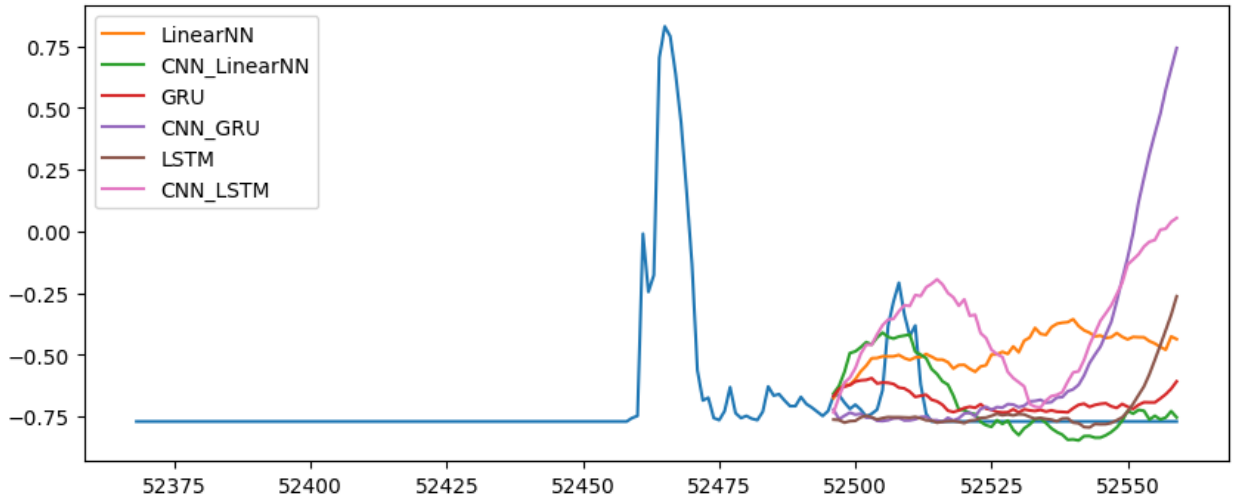

Рис.8 DMS передбачення на 64 кроки моделей датасету solar AL

Цікавою особливістю датасету solar AL є те що при передбачені на довгий проміжок часу (16, 64) найкращої точності досягнула LinearNN IMS модель, при тому що зазвичай IMS метод погано працює при великих T.

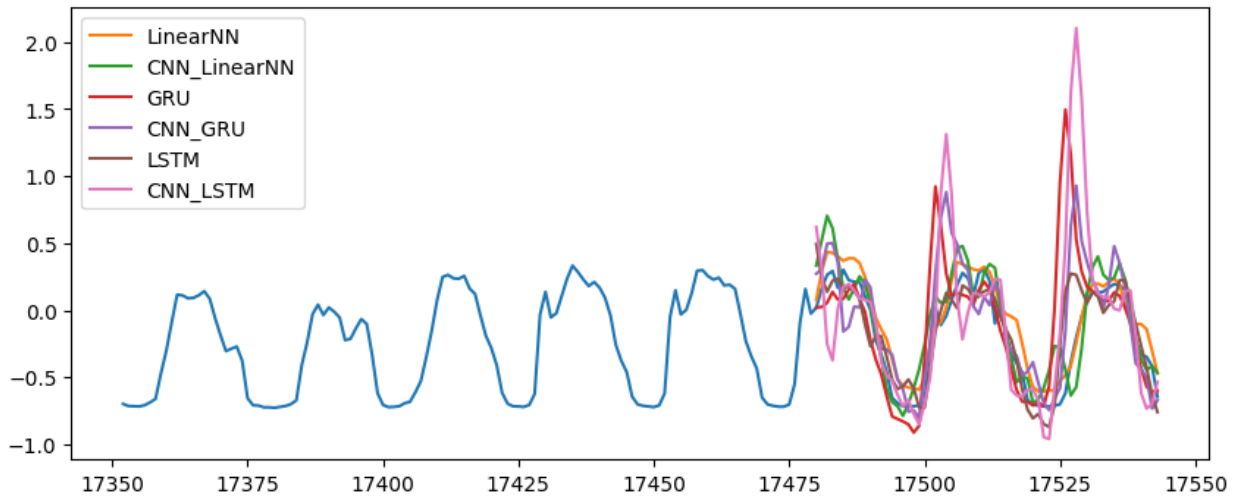

Рис.9 DMS передбачення на 64 кроки моделей датасету traffic

traffic. На цьому датасеті CNN GRU DMS мала найкращу точність поміж інших моделей на всіх T. Також можна помітити що рекурентні моделі що використовують згортковий шар мають кращу точність порівняно з аналогами що не мають його.

Висновки

В цій роботі було протестовано шість архітектур нейронних мереж на трьох різних датасетах. З експериментів проведених в цій роботі стає зрозуміло, що жодна з моделей не є універсальним рішенням для передбачення будь-яких часових рядів. Точність моделі сильно залежить від природи використаних даних.

Повнозв'язна нейронна мережа показала непоганий результат, особливо при передбачені далеко в майбутнє та використанні ітеративного методу.

Використання згорткового шару може сильно покращити точність моделі, хоча в тандемі лише з лінійним шаром та ітеративним методом може робити серйозні помилки, що сильно погіршить точність.

Рекурентні архітектури такі як GRU та LSTM показали чудові результати, хоча в деяких випадках все ще програють більш простішим моделям.

Список літератури

1. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. 6 Dec 2017, arXiv:1706.03762
2. Ailing Zeng, Muxi Chen, Lei Zhang, Qiang Xu. Are Transformers Effective for Time Series Forecasting? 17 Aug 2022, arXiv:2205.13504
3. Hochreiter S, Schmidhuber, J"urgen. Long short-term memory. Neural computati-on. 1997;9(8):1735–80.
4. Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. Learning Phrase Representati-ons using RNN Encoder-Decoder for Statistical Machine Translation. 3 Sep 2014, arXiv:1406.1078
5. Souhaib Ben Taieb, Rob J Hyndman, et al. Recursive and direct multi-step forecasting: the best of both worlds, volume 19. Citeseer, 2012
6. Guillaume Chevillon. Direct multi-step estimation and forecasting. Journal of Economic Surveys, 21(4):746–785, 2007
7. Hewamalage, H., Ackermann, K., Bergmeir, C. Forecast evaluation for data scientists: common pitfalls and best practices. Data Min Knowl Disc 37, 788–832 (2023)
8. Alexander Rehmer, Andreas Kroll, On the vanishing and exploding gradient problem in Gated Recurrent Units, IFAC-PapersOnLine, Volume 53, Issue 2, 2020, Pages 1243-1248, ISSN 2405-8963
9. Ant´onio H. Ribeiro, Koen Tiels, Luis A. Aguirre, Thomas Sch¨on. Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR 108:2370-2380, 2020.
10. [Електронний ресурс] Multivariate Time series Data sets URL: https://github.com/laiguokun/multivariate-time-series-data

11. Manaswi, N.K. (2018). RNN and LSTM. In: Deep Learning with Applications Using Python. Apress, Berkeley, CA, 115-126
12. Cootner, Paul H. (1964). The random character of stock market prices. MIT Press. ISBN 978-0-262-03009-0
13. [Електронний ресурс] RNN, LSTM & GRU. URL: http://dprogrammer.org/rnn-lstm-gru
14. Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. 30 Jan 2017, arXiv:1412.6980v9