Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти____другий (Магістерський)_____
(рівень вищої освіти)

Система регіонального прогнозування погоди на базі технології штучного інтелекту
(тема)

Виконав: здобувач 2024 року навчання, групи

СКСм-23-2_____

_____Довгаль В.Р._____
(прізвище, ініціали)

Спеціальність 123 _ Комп'ютерна інженерія

Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_____

_____Спеціалізовані комп'ютерні системи__
(повна назва освітньої програми)

Керівник роботи ____проф. Немченко В.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Чумаченко С.В
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проектування обчислювальної техніки_____

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Комп'ютерна інженерія _____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«__»_____ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Довгалю Владиславу Романовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Система регіонального прогнозування погоди на базі технології штучного інтелекту

затверджена наказом по університету від "08 "_____11_____2024 р. № 1189Ст_____

2. Термін подання здобувачем роботи до екзаменаційної комісії _____20.01.2025_____

3. Вихідні дані до роботи (проекту)_____

PyCharm, Visual Studio Code, Arduino Nano 33 BLE Sense_____

Мови програмування Python, C++_____

_____

4. Перелік питань, що потрібно опрацювати у роботі _____

Аналіз предметної області та постановка завдання

Дослідження актуальний нейронних мереж

Розробка нейронної мережі та аналіз результатів тестування

Розробка програмного забезпечення апаратної частини

_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 18 слайдів_____

_____

_____

_____

_____

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 02.09.2024

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження теми	02.09.2024-30.09.2024	
2	Вивчення основ метеорології та штучного інтелекту	01.10-2024-15.10.2024	
3	Дослідження актуальних робіт у сфері прогнозування погоди	16.10.2024-07.11.2024	
4	Визначення архітектури нейронної мережі	08.11.2024-22.11.2024	
5	Реалізація та тестування програмного забезпечення моделі, аналіз результатів	29.11.2024-10.12.2024	
6	Розробка програмного забезпечення апаратної частини	11.12.2024-18.12.2024	
7	Оформлення пояснювальної записки	18.12.2024-20.01.2025	
8	Захист проекту	20.01.2025-24.01.2025	

Здобувач _____
(підпис)

Керівник роботи (проекту) _____ Проф.Немченко В.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 56 сторінок, 19 рисунків, 23 джерел за переліком посилань.

ПРОГНОЗУВАННЯ ПОГОДИ, МЕТЕОРОЛОГІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, MULTI-LAYEREDPERCEPTRON, BACKPROPAGATION, SGD, LEAKYRELU, DROPOUT, BATCHNORM, LAYERNORM, PYTHON, ERA5,TENSORFLOW, ARDUINO, C++, TFLITE, ELOQUENTTINYML

Метою кваліфікаційної роботи є дослідження можливостей використання штучного інтелекту для прогнозування погоди з високою точністю та ефективністю. У процесі роботи розглянуто архітектури нейронних мереж, методи навчання та оптимізації моделей для забезпечення точних прогнозів.

Під час дослідження була встановлена найкраща для задач прогнозування погоди архітектура нейронної мережі. Проаналізовані методи навчання моделі та вибраний найкращий метод. Також проведено аналіз методів активації, технік нормалізації та регуляризації.

Процес створення моделі включав отримання через API даних. Досліджено ефективність використання вищезазначених даних для навчання та валідації моделі, а також побудовано графіки для аналізу результатів.

Зокрема, проведено розробку програмного забезпечення для апаратної частини, що дозволило інтегрувати раніше навчену нейронну мережу з реальними даними.

Результатом цієї роботи стала ефективна та перспективна для прогнозування погоди архітектура, а також продемонстрована взаємодія апаратної частини з моделлю нейронної мережі.

ABSTRACT

The explanatory note contains 56 pages, 19 figures, and 23 references as listed.

PREDICTION OF WEATHER, METEOROLOGY, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, NEURAL NETWORK, MULTI-LAYERED PERCEPTRON, BACKPROPAGATION, SGD, LEAKY RELU, DROPOUT, BATCHNORM, LAYERNORM, PYTHON, ERA5, TENSORFLOW, ARDUINO, C++, TFLITE, ELOQUENTTINYML

This qualification work aims to explore the possibilities of using artificial intelligence for high accuracy and efficiency in weather forecasting. Throughout the work, architectures of neural networks, methods of training, and model optimization have been considered to ensure precise predictions.

During the research, the best architecture for weather forecasting tasks was established. The model training methods were analyzed, and the most suitable method was selected. Additionally, an analysis of activation, normalization, and regularization methods was carried out.

The model creation process included obtaining data through APIs. The effectiveness of using the data mentioned above for training and validating the model was investigated, and graphs were built for result analysis.

In particular, software development for the hardware part was conducted, allowing the integration of the previously trained neural network with actual data.

The result of this work is an effective and promising architecture for weather forecasting, as well as a demonstrated interaction between the hardware part and the neural network model.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

ШНМ – Штучні Нейронні Мережі

ML – Machine Learning

GFS – Global Forecast System

AIFS – AI Forecasting System

IFS – Integrated Forecasting System

HRES – High Resolution Forecast

ECMWF – European Centre for Medium-Range Weather Forecasts

RMSE – Root Mean Square Error

MAE – Mean Absolute Error

MLP – Multilayer Perceptron

GD – Gradient Descent

ReLU – RectifiedLinear Unit

Tanh – Hyperbolic Tangent

IDE – Integrated DevelopmentEnvironment

PIP – Python Installer Package

API – Application Programming Interface

ВСТУП

Сучасний світ генерує величезні обсяги даних, які можуть бути використані для вдосконалення процесів у різних сферах життя. Однак обробка й аналіз цих даних стають дедалі складнішими задачами, що вимагають застосування інноваційних підходів. Технології штучного інтелекту (ШІ) відіграють ключову роль у вирішенні цих викликів, оскільки вони здатні автоматизувати обробку великих обсягів інформації, знаходити приховані закономірності та забезпечувати точне прогнозування.

ШІ вже сьогодні змінює наше уявлення про можливості технологій, допомагаючи вирішувати завдання, які раніше здавалися надзвичайно складними. У медицині алгоритми аналізують знімки для ранньої діагностики захворювань, створюють персоналізовані плани лікування та навіть допомагають під час роботизованих операцій. У сфері аналітики ці технології дозволяють ефективно працювати з великими обсягами даних, виявляти закономірності, оптимізувати бізнес-процеси та прогнозувати попит. У військовій промисловості ШІ використовується для управління сучасними системами, такими як безпілотні апарати, підвищуючи точність і ефективність операцій, а також знижуючи ризики для людей.

Ця робота спрямована на дослідження можливостей ШІ у сфері прогнозування погоди. В умовах глобальних змін клімату та збільшення частоти екстремальних погодних явищ виникає нагальна потреба в розробці ефективних систем прогнозування, здатних забезпечити високий рівень точності прогнозів.

У роботі буде розглянуто штучний інтелект з нуля, включаючи його навчання, ефективне застосування, взаємодію з апаратною частиною. Особливу увагу буде приділено різновидам нейронних мереж, процесам їх навчання, оптимізації та тестування, а також опису системи на базі Arduino, здатної отримувати реальні показники та порівнювати їх з результатами прогнозування реалізованої моделі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

В цьому розділі проводиться аналіз основ штучного інтелекту та метеорології, що є фундаментом для подальшої роботи з прогнозування погоди за допомогою нейронних мереж. Окремо розглядаються ключові поняття і методи ШІ, такі як машинне навчання, нейронні мережі та їх основні типи, а також їх застосування в різних сферах науки і техніки. Аналізуються основи метеорології, які забезпечують теоретичну базу для розуміння процесів атмосферних явищ та їх прогнозування. Також проаналізовані сучасні рішення в цій сфері, сформована мета та постановка завдання дослідження.

1.1 Основи штучного інтелекту

Штучний інтелект (ШІ) [1] — це галузь комп'ютерних наук, яка займається створенням інтелектуальних систем, здатних виконувати завдання, що вимагають людської інтелектуальної діяльності. Штучний інтелект є потужним інструментом, який пропонує величезний потенціал для поліпшення життя та вирішення складних завдань. Однією з ключових областей ШІ є машинне навчання — метод аналізу даних, який дозволяє комп'ютерним системам самостійно навчатися та вдосконалюватися без явного програмування.

Машинне навчання [2] — це підгалузь ШІ, яка дозволяє комп'ютерним системам навчатися на основі досвіду та даних. За допомогою машинного навчання комп'ютерні програми можуть самостійно покращувати свою продуктивність і адаптуватися до нових завдань. Машинне навчання включає кілька підгалузей:

- Навчання з учителем (Supervised Learning) — алгоритми, які навчаються на помічених даних, тобто на таких, що містять вхідні значення та відповідні їм правильні виходи. Метою є навчити модель правильно передбачати вихід за новими, невідомими даними.

Приклад використання: Розпізнавання електронних листів як спам або не спам. У цьому випадку модель навчається на даних, де кожен лист позначений як спам або не спам, і потім застосовує ці знання для класифікації нових листів.

- Навчання без учителя (Unsupervised Learning) — алгоритми, які працюють із неструктурованими даними, де немає вказівок на правильні результати. Такі алгоритми використовуються для виявлення прихованих патернів або груп у даних, наприклад, для кластеризації.

Приклад використання: Кластеризація клієнтів за схожістю в покупках для маркетингових кампаній. Моделі навчаться виявляти групи клієнтів із подібними інтересами без необхідності знати конкретні категорії заздалегідь.

- Навчання з підкріпленням (Reinforcement Learning) — метод, де агент навчається приймати рішення через взаємодію з навколишнім середовищем. Він отримує винагороду або покарання за кожну дію, що дозволяє йому оптимізувати своє поведінку для досягнення певної мети.

Приклад використання: Навчання робота ходити або грати в гру, таку як шахи чи го. Робот отримує винагороду за кожен правильний крок або хід, поступово покращуючи свою стратегію для досягнення кращих результатів.

Машинне навчання дозволяє ШІ автоматично обробляти великі обсяги даних, розпізнавати образи, аналізувати тексти та приймати рішення.

Однак, глибоке навчання (Deep Learning) [3] — це ще більш потужний крок у розвитку штучного інтелекту. Це підгалузь машинного навчання, яка використовує штучні нейронні мережі (ШНМ). Нейронні мережі (NeuralNetworks) — це модель, натхнена роботою людського мозку. Вони складаються з вузлів, які називаються нейронами, що з'єднані між собою різними рівнями, або шарами. Це дозволяє мережі обробляти вхідні дані та виявляти закономірності. Шари бувають: вхідними (отримують вхідні дані для

обробки), прихованими (виконують основну обробку даних) та вихідними (повертають прогнозоване значення або класифікацію). У простих нейронних мережах може бути лише кілька шарів, тоді як у складних — кілька рівнів, що дозволяють ефективно вирішувати складні завдання, як, наприклад, розпізнавання зображень (наприклад, ConvolutionalNeuralNetworks, CNN)або обробка природної мови (наприклад, RecurrentNeuralNetworks, RNN). Кожен шар нейронної мережі складається з кількох нейронів, і кожен нейрон у шарі зв'язаний з нейронами в наступному шарі. Це дозволяє кожному шару обробляти інформацію і передавати її далі для більш детальної обробки, що дозволяє досягати високої точності при розв'язанні конкретних задач. Ця структура, яка включає кількість шарів, нейронів у кожному шарі та їхні зв'язки, називається архітектурою[4] нейронної мережі.

Навчання нейронної мережі — це процес навчання нейронної мережі виконанню певного завдання. Нейронні мережі навчаються шляхом попередньої обробки кількох великих наборів розмічених або нерозмічених даних. На основі цих прикладів мережі можуть більш точно обробляти невідомі вхідні дані.

Функція активації являє собою нелінійне перетворення, яке поелементно застосовується до вхідних даних. Вона додається до штучної нейронної мережі, щоб допомогти мережі вивчити складні закономірності в даних. Порівняння з моделлю, основаною на біологічних нейронах у мозку, показує, що функція активації визначає, які нейрони будуть активовані наступними. Функція втрат вимірює, наскільки добре передбачення моделі відповідають справжнім значенням.

Оптимізація нейронної мережі — це процес налаштування параметрів моделі (ваг і зміщень), щоб мінімізувати помилку, яка вимірюється функцією втрат, і зробити мережу більш точною у передбаченнях. Цей процес включає алгоритми, методи та стратегії для ефективного навчання моделі.

1.2 Основи метеорології

Метеорологія[5] – це наука, яка вивчає атмосферні явища та процеси, що визначають погодні умови на Землі. Вона включає в себе дослідження та аналіз температури, тиску, вологості, вітру, опадів і інших атмосферних факторів, а також їх взаємодію та вплив на кліматичні умови. Ця наука має важливе значення для забезпечення безпеки людей і оптимізації діяльності в багатьох галузях:

- Захист життя та майна: Прогнози погоди допомагають уникати небезпечних наслідків стихійних явищ, таких як урагани, повені, грози чи екстремальні температури.

- Сільське господарство: Інформація про погодні умови дозволяє фермерам планувати посіви, збір врожаю та іригацію, зменшуючи втрати через несприятливі умови.

- Транспорт: Авіація, судноплавство та наземний транспорт покладаються на точні прогнози погоди для забезпечення безпеки та ефективності перевезень.

- Енергетика: Дані метеорології використовуються для управління відновлюваними джерелами енергії, такими як вітрова та сонячна, а також для планування споживання енергоресурсів.

- Кліматологія та екологія: Розуміння атмосферних процесів дозволяє аналізувати довгострокові кліматичні тенденції, боротися зі зміною клімату та захищати екосистеми.

- Щоденне життя: Прогнози допомагають людям планувати свій день, роботу чи відпочинок, враховуючи погодні умови.

Штучний інтелект є основним інструментом для складання прогнозів погоди в сучасних умовах. Відтак, головним джерелом усієї прогностичної інформації є глобальні чисельні моделі атмосфери. Це програми, які відтворюють всю атмосферу Землі, починаючи від поверхні і закінчуючи верхніми шарами стратосфери. Основним джерелом даних для цих моделей є супутникові спостереження в усіх видимих та невидимих спектральних діапазонах, а також дані з кораблів, літаків та аерологічних зондів. Дані з наземних станцій вже не мають такого сильного впливу на якість прогнозів, хоча вони й досі використовуються.

Однак, для проведення досліджень у цій сфері або виконання практичних завдань зовсім не обов'язково мати у своєму розпорядженні власний радар чи супутник. Цілком достатньо простого мікроконтролера, оснащеного необхідними датчиками, наприклад, Arduino Nano 33 BLE Sense [6]. Такий пристрій дозволяє отримувати базові дані зовнішнього середовища, такі як температура, вологість тощо. Ці дані можна обробляти, використовувати для навчання ШІ або для реалізації інших завдань, залежно від потреб дослідника чи практичного проєкту.

Ще одна важлива деталь для досліджень у цій сфері, особливо пов’язаних зі штучним інтелектом, – це готові набори даних, які спрощують процес аналізу та розробки моделей. Звісно, можна зібрати власний набір даних, однак цей процес займає дуже багато часу, і чим більший обсяг даних, тим точнішими будуть результати досліджень. Популярними наборами даних єERA5, GHCN, GFS, MODIS, тощо.

1.3Огляд сучасних рішень

Довгий час машинне навчання здебільшого використовувалося для постобробки сирих даних. Ніхто не намагався використовувати нейронні мережі для створення повноцінних моделей прогнозування погоди. Перші тести на мережах з низьким розділенням показали, що створення таких систем

не є нездійсненною задачею. До сучасних моделей можна віднести: FourCastNetv2-small від Nvidia, GraphCast від Google, Pangu-Weather від Huawei та AIFS від ECMWF. Далі їх короткий опис:

- FourCastNetv2-small (FourCastNet): наступна ітерація системи глибокого навчання FourCastNet, розроблена компанією Nvidia та її співробітниками. У цій моделі використовуються сферичні нейронні оператори Фур'є для врахування просторових залежностей. Ця модель є зменшеною версією, яка може поміститися в один накопичувач Nvidia A100 40 ГБ для проведення обчислень. Вона була навчена за допомогою ERA5 для мінімізації середньоквадратичної помилки прогнозу та працює при 0,25°.

- Graphcast[7]: система на основі глибокого навчання, розроблена Google Deepmind. У ній використовується архітектура графової нейронної мережі з структурою кодер-процесор-декодер з багатосітковим поданням. Модель була навчена на реаналізі ERA5 з роздільною здатністю 0,25° і налаштована на прогнозі ECMWF HRES з метою мінімізації середньоквадратичної помилки прогнозу.

- Pangu-Weather: система на основі глибокого навчання, розроблена компанією Huawei. Вона використовує архітектуру 3D-трансформера для врахування просторових залежностей і складається з кількох моделей, що дозволяють робити прогнози на різних тимчасових інтервалах (наприклад, 24 години, 6 годин). Вона була навчена за допомогою ERA5 для мінімізації середньоквадратичної помилки прогнозу для кожної моделі та працює при 0,25°.

- AIFS: система на основі глибокого навчання, розроблена в ECMWF. Використовуються графові нейронні мережі. Навчання відбувалося на основі реаналізу ERA5. AIFS має 13 рівнів тиску, працює з роздільною

здатністю близько 1 градуса і дозволяє прогнозувати вітер, температуру, вологість і геопотенціал. На поверхні AIFS робить прогнози для температури на 2 м, вітру на 10 м, приземного тиску тощо. AIFS була навчена мінімізувати середню квадратичну помилку.

Чітко видно, що у всіх моделей є одна спільна особливість – вони всі навчалися на реаналізі ERA5. Реаналізи – це цифрові архіви погодної інформації. Вони дозволяють вивчати погоду навіть у тих місцях, де рідко є мережа метеостанцій. В певному сенсі можна сказати, що реаналіз – це прогін гідродинамічної моделі з нульовим заздалегідь прогнозом, а оскільки модель ECMWF є найкращою у світі, то й їхній реаналіз є еталонним. Саме з цієї причини ERA5 частіше всього використовувався для навчання нейронних мереж.

Актуальні тести продемонстрували, що нові нейромережеві моделі досягли вражаючих результатів у прогнозуванні погоди. Нові моделі, такі як FourCastNet і Pangu-Weather, показали точність, яка порівнянна з результатами гідродинамічної версії IFS від ECMWF, що протягом багатьох років була еталонною моделлю для прогнозування погоди, розробленою Європейським центром середньострокових прогнозів. Проте моделі GraphCast і AIFS змогли суттєво перевершити цей стандарт. Класична детермінована версія ECMWF прогнозує температуру на рівні 850 hPa (1500 метрів) на 10 діб з точністю 36-37%. Машинне навчання підвищує точність до 45%. Зростання якості спостерігається на всіх строках заздалегідь. Баричне поле на 8-10 діб нова нейромережна модель прогнозує на 5-8% точніше, ніж звичайна версія ECMWF.

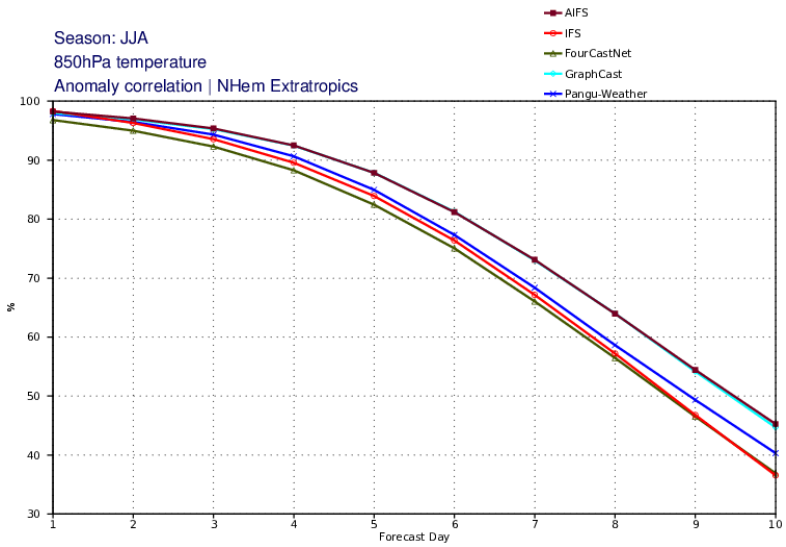

Рисунок 1.1 – Якість прогнозування температури на рівні 850 hPa (1500 м)

Детальний аналіз результатів прогнозування температури на рівні 850 hPa (1500 метрів) показав, що AIFS і GraphCast мають майже ідентичні показники точності. Цікаво, що AIFS з'явилася пізніше, ніж модель від Google, і використовує ту ж саму архітектуру, яку розробили в GraphCast.

Однак, за результатами деяких досліджень, явним фаворитом виступає GraphCast. Самі розробники цієї моделі у своїй науковій статті пишуть, що «GraphCast має вищі показники прогнозування погоди, ніж HRES (ECMWF), при оцінці 10-добових прогнозів з горизонтальним розрізненням 0,25° по широті/довготі та на 13 вертикальних рівнях».

HRES (High-Resolution Forecast) — цечастинагідродинамічноїмоделі IFS (Integrated Forecast System) в ECMWF.

«GraphCast має кращі показники на всіх часових інтервалах, з покращенням на 7%-14%. Він перевершив HRES за 90,3% з 1380 завдань, причому значно ($p \leq 0,05$, номінальний обсяг вибірки $n \in \{729, 730\}$), а також перевершив HRES за 89,9% завдань. При виключенні рівня 50 гПа GraphCast значно перевищує HRES за 96,9% з 1280 завдань. При виключенні рівнів 50 і 100 гПа GraphCast перевершує HRES на 99,7% з 1180 завдань. При оцінці по регіонах результати залишаються стабільними по всьому світу. Ми також порівняли продуктивність GraphCast з найкращою конкуренцією ML-моделлю погоди

Pangu-Weather і виявили, що GraphCast перевершує її на 99,2% з 252 представлених завдань», — зазначають у статті.

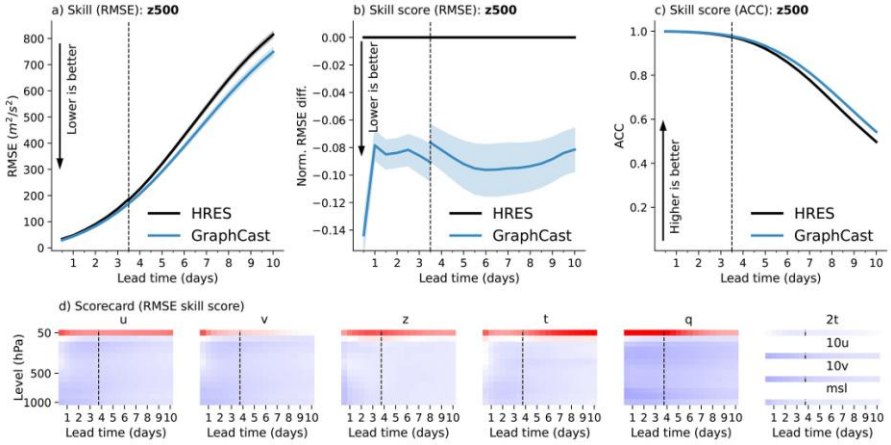

Рисунок 1.2 – Перевага GraphCast над HRES від ECMWF

1.4 Постановка завдання

Метою кваліфікаційної роботи є вивчення технології штучного інтелекту, дослідження різновидів нейронних мереж та методологій оцінювання їх роботи.

Об'єктом дослідження є технологія штучного інтелекту та його практичне застосування для створення системи прогнозування погоди з високою точністю.

Предметом дослідження є архітектури нейронних мереж для прогнозування погоди, методи їх навчання та регуляризації даних, оптимізація моделей, а також показники ефективності прогнозів, отриманих за допомогою штучного інтелекту.

На практиці досліджені найпопулярніші моделі мереж, описані етапи їх ініціалізації, навчання, приведені приклади прогнозування. Оцінені результати роботи моделей. Як єдиний параметр моделей, використовується температура повітря. Зокрема, розглянута апаратна частина, яка вимірює температуру повітря в реальному часі та порівнює отримані значення з прогнозованими даними нейроної мережі.

2 ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПРОГНОЗУВАННЯ ПОГОДИ

Цей розділ присвячений аналізу штучних нейронних мереж (ШНМ), що використовуються для прогнозування погоди. Особливий акцент зроблено на прогнозуванні температури. Аналіз включає порівняння продуктивності штучних нейронних мереж на основі середньоквадратичної помилки (Root Mean Square Error,RMSE). Основні цілі цього розділу - відповісти на ключові питання, такі як: Яка архітектура нейронних мереж є найбільш ефективною для прогнозування погоди? Які методи навчання та оптимізації нейронних мереж найбільш ефективні? Які функції активації найкраще підходять для прогнозування погоди?Які є методи регуляризації та нормалізації даних?

2.1Наукові роботи у сфері прогнозування погоди

Існує багато типів нейронних мереж, які класифікуються за різними критеріями, включаючи структуру, потік даних, щільність та тип використовуваних нейронів, глибину, шари та їхні фільтри активації тощо. Розберемо деякі з них, які відрізнились високими показниками:

2.1.1 Модель 3-9-1

Модель описана в роботі «Forecasting Maximum Seasonal Temperature Using Artificial Neural Networks: Tehran Case Study»E. FahimiNezhad, G. FallahGhalhari, F. Bayatan(2019) [8]. Головною метою цього дослідження є прогнозування максимальної температури за допомогою методів нейронних мереж. У даній роботі використовувалися середньомісячні значення максимальної температури, різниця між максимальною і мінімальною

температурами, сонячні години, швидкість вітру, середня відносна вологість та середні температури синоптичної станції Тегеран Мехрабад за статистичний період з 1951 по 2010 рік для прогнозування максимальної температури в Тегерані взимку. Для цього 70% даних було виділено для навчання мережі, а 30% даних були присвячені тестуванню та валідації.Як показано на рисунку2.1, процес максимальної температури в Тегерані є нелінійним. Тому використання статистичних моделей для прогнозування лінійних процесів неможливе. Для уникнення цієї проблеми, краще використовувати моделі, які мають здатність прогнозувати нелінійні процеси.

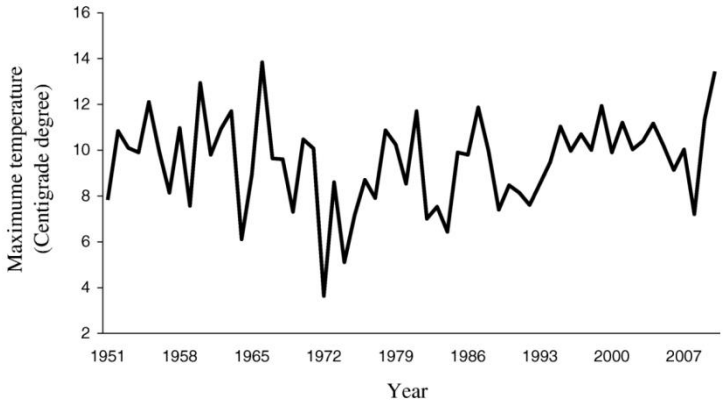

Рисунок 2.1 – Графік часових даних максимальної температури в Тегерані взимку за статистичний період з 1951 по 2010 рік

Автори роботи роблять висновок, що найбільшпідходящою структурою нейронної мережі для прогнозування максимальної зимової температури у Тегерані є модель з трьома нейронами у вхідному шарі, прихованим шаром з 9 нейронами та використанням гіперболічної тангенсової функції(функція активації) у прихованому та вихідному шарах. Тобто це структура 3-9-1. Результати її використання показали, що середньоквадратична помилка, коефіцієнт кореляції та середня абсолютна помилка для етапів навчання та тестування становлять відповідно 0.001, 0.997, 0.61 і 0.104, 0.997, 0.311.

2.1.2 Модель 6-6-1

Модель описана в роботі «Analysis of artificial neural network performance based on influencing factors for temperature forecasting applications»M. Madhiarasan, M. Tipaldi, P. Siano(2020) [9].У цій роботі описано штучну нейронну мережу прямого поширення з алгоритмом навчання зворотного розповсюдження: помилка на виході обчислюється і поширюється назад на вхідний шар через приховані шари.Для навчання використовується реальні дані, які збиралися з Національного управління океанічних і атмосферних досліджень (США) з січня 2016 року по грудень 2018 року, 70% для навчання та 30% для тестування.Під час фази зворотного поширення ваги ШНМ регулюються за допомогою методу градієнтного спуску (оптимізаційний алгоритм).

Вхідний шар складається з 6 вхідних параметрів, пов'язаних із застосуванням прогнозування температури (див. рисунок 2.2). Кількість прихованих нейронів у прихованому шарі – від 1 до 30. Вихідний шар має один вихідний нейрон, тобто прогнозовану температуру. Гіперболічна тангенсова сигмоїдна функція активації використовується як у прихованих шарах, так і у вихідному шарі.Помилка на виході ШНМ отримується шляхом обчислення різниці між обчисленими та цільовими значеннями температури (функція втрат, lossfunction), останні надаються у наявному наборі даних.

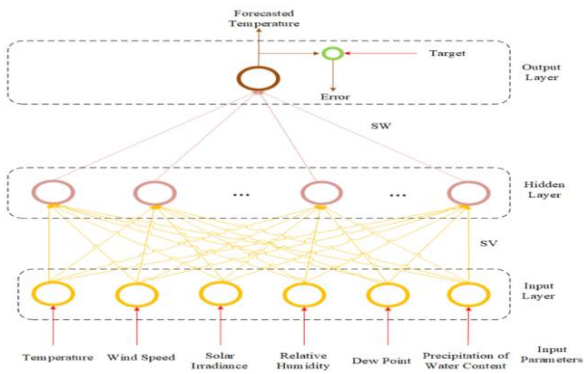

Рисунок 2.2 – Архітектура моделі 6-6-1

Згадується також і про важливість виміру прихованого шару шляхом вибору належної кількості прихованих нейронів:недопідгонка може статися у випадку занадто малої кількості прихованих нейронів, тоді як велика кількість прихованих нейронів може спричинити перенавчання та повільну збіжність під час процесу навчання ШНМ.При цьому, кількість визначаєтьсяабо емпіричним шляхом, або “за допомогою специфічних математичних формулювань та евристичних технік”.Методом спроб було отримано, що найкраща кількість прихованих нейронів у приховану шарі – це 6. Тобто це структура 6-6-1.Результати її використання показали, що середньоквадратична помилка, коефіцієнт кореляції та середня абсолютна помилка для етапів навчання становлять відповідно 0.0059, 1, 0.0164, а для етапів тестування — 0.0770, 1, 0.0728.

2.1.3GraphCast

Модель наведена в роботі «GraphCast: Learnings killful medium-range global weather fore casting»RemiLam, Alvaro Sanchez-Gonzalez, Matthew Willson та ін. (2022) [7].У цій роботі використано загальнодоступні дані Європейського центру Середньострокове прогнозування (ECMWF) ERA5, який є великим масивом данихз 1959 року до сьогодні.Всі вхідні дані нормалізовані. Для кожної фізичної змінної ми обчислено середнє значення та стандартне відхилення за рівнями тиску за період 1979–2015 років, і використано ці значення для нормалізації до нульового середнього та одиничної дисперсії.

GraphCast використовує складну GNN (GraphNeuralNetwork) архітектуру в конфігурації “кодування-обробка-декодування” або “encoder-processor-decoder”. Система бере вхідні дані, такі як температура і тиск, обробляє їх у багатошаровій сітці (мульти-сітка), а потім повертає результати назад на початкову сітку. Вхідні дані включають погодні параметри, які представлені у

вигляді вузлів на сітці, де кожен вузол представляє вертикальний зріз атмосфери в конкретній точці широти-довготи. Мульти-сітка складається з кількох рівнів вузлів і ребер, які дозволяють обробляти дані з різним ступенем деталізації. Ребра сітки з'єднують вузли і допомагають передавати інформацію між ними, що дозволяє системі враховувати довгострокові взаємодії між різними частинами атмосфери. Процесор виконує багаторазові раунди передачі повідомлень по мульти-сітці, обробляючи та оновлюючи інформацію на кожному рівні, що дозволяє ефективно поширювати інформацію по всій сітці. Декодер бере оброблену інформацію з мульти-сітки і повертає її назад на початкову сітку для отримання кінцевого прогнозу погоди. З'єднання між вузлами сітки (Grid2Mesh і Mesh2Grid) використовуються для передачі інформації в обох напрямках, забезпечуючи повну взаємодію між вузлами.

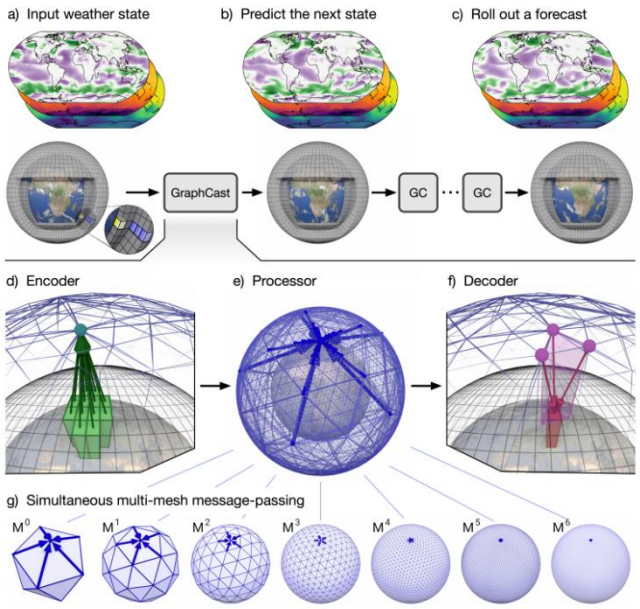

Рисунок 2.3 – Архітектура моделі GraphCast

На всіх рівнях використовуються багатошарові перцептрони (MLP) з одним прихованим шаром, розмірами прихованих і вихідних шарів - 512 нейронів (за винятком останнього шару декодера, який має вихідний розмір 227, що відповідає кількості прогнозованих змінних для кожного вузла сітки). Обрана також функція активації “swish” для всіх MLP. Усі MLP доповнюються шаром нормалізації (LayerNorm), за винятком MLP декодера.

Для покращення здатності моделі робити точні прогнози на більш ніж один крок, використовується авторегресійний режим навчання, де передбачуваний наступний крок моделі підключається як вхід для прогнозування наступного кроку. Остаточна версія GraphCast була навчена на 12 авторегресійних кроках, дотримуючись плану навчання, описаного далі. Процедура оптимізації обчислювала втрати на кожному кроці прогнозу відповідно до відповідного реального значення, а градієнти помилок щодо параметрів моделі розповсюджувалися назад через всю розкручену послідовність ітерацій моделі (тобто використовувалася зворотне розповсюдження помилки через час). Для оптимізації використовується метод градієнтного спуску.

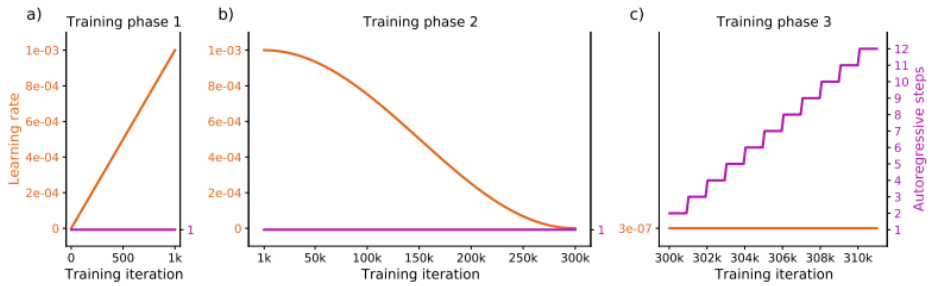

Рисунок 2.4 – Етапи тренування GraphCast

Результати роботи показують, що GraphCast:<2018 та GraphCast:<2019 мають нижчі навички у порівнянні з HRES на ранніх етапах попередження для оцінки 2021 року. Однак для інших змінних GraphCast:<2018 та GraphCast:<2019 набагато перевершує HRES.

2.2 MLP

Проаналізувавши численні дослідження та наукові праці, можна дійти висновку, що багатошаровий перцептрон[10] (Multi-LayeredPerceptron або MLP) є найкращим вибором для прогнозування часових рядів. Розглянуті моделі, такі як 3-9-1, 6-6-1 та GraphCast, показали високу ефективність та

точність у прогнозуванні різних параметрів за допомогою методів MLP. У цій частині розглянемо основну теорію та переваги цієї моделі.

Багатошаровий перцептрон — це клас штучних нейронних мереж прямого поширення, що складаються як мінімум з трьох шарів: вхідного, прихованого та вихідного. За винятком вхідних, всі нейрони використовують нелінійну функцію активації. Під час навчання багатошарового перцептрона (MLP) використовується навчання з учителем. Як функції активації нейронів використовуються сигмоїдальні функції: логістична або гіперболічний тангенс.

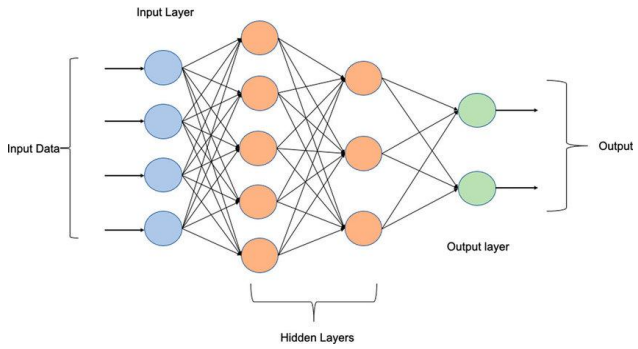

Рисунок 2.5 – Приклад архітектури MLP

Ключові переваги:

- Нелінійність: Як зазначено в описі моделі 3-9-1, зміна температури часто відбувається нелінійно. MLP може обробляти складні нелінійні взаємозв'язки у даних. Прогнозування погоди включає в себе багато таких взаємозв'язків між різними атмосферними параметрами, і MLP може їх враховувати.

- Можливості навчання: MLP може бути навчена на різних тимчасових масштабах, що дозволяє робити короткострокові та довгострокові прогнози. Алгоритм навчання зворотного розповсюдження добре себе показав в описі моделі 6-6-1. Окрім того, використання MLP з авторегресійним

режимом навчання дозволяє моделі робити точні прогнози на кілька кроків вперед. Це особливо корисно для середньострокових прогнозів погоди.

- Обробка великого обсягу даних: MLP може ефективно обробляти великі обсяги даних, що є важливим для точного прогнозування погоди, яке вимагає аналізу значної кількості історичних даних, наприклад, ERA5 від ECMWF.

2.3 Алгоритм зворотного поширення помилки та градієнтний спуск

Як вже було зазначено, модель багатошарового перцептрона (MLP) навчається з учителем. Розглянемо найпопулярніший метод навчання, який найчастіше використовуються в розглянутих моделях нейронних мереж.

Алгоритм зворотного поширення помилки (Backpropagation) [11] — це метод навчання штучних нейронних мереж з учителем, який використовується для оптимізації вагів, мінімізуючи помилки прогнозу. Зворотне поширення складається з двох основних фаз: пряме поширення (forward propagation) та зворотне поширення помилки (backward propagation).Зазвичай використовується разом із градієнтним спуском [12] (алгоритм оптимізації). Основна мета зворотного поширення помилки — обчислення градієнтів функції втрат щодо вагів і зміщень, а метод градієнтного спуску використовує ці градієнти для оновлення параметрів моделі, щоб зменшити помилку.

Градієнтний спуск — це метод знаходження мінімального значення функції втрат.Основні види градієнтного спуску:

- Стохастичний градієнтний спуск (StochasticGD): Виконує оновлення параметрів моделі після кожного окремого прикладу навчання, обраного випадковим чином. Це дозволяє швидко оновлювати ваги, але може бути менш стабільним через високу варіативність. Швидко навчається, особливо корисне для великих наборів даних.

- Пакетний градієнтний спуск (Batch GB): Використовує весь навчальний набір даних для одного оновлення параметрів моделі. Кожне оновлення враховує інформацію з усіх прикладів навчання, що забезпечує стабільне наближення до мінімуму. Стабільні оновлення ваг та плавне зменшення втрат.

- Міні-пакетний градієнтний спуск (Mini-batchGB): Цей метод полягає у виборі невеликих груп прикладів навчання (міні-пакетів) для кожного оновлення параметрів моделі. Це дозволяє збільшити ефективність обчислень, порівняно з обробкою кожного прикладу окремо або всього набору даних одразу.Поєднує стабільність пакетного градієнтного спуску з ефективністю стохастичного градієнтного спуску.

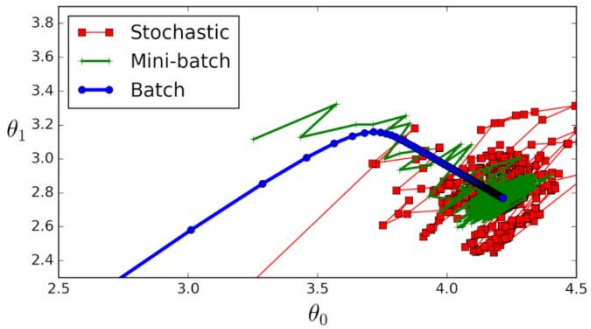

Рисунок 2.6 – Результативність видів градієнтного спуску

Схема роботи зворотного поширення помилки з використанням градієнтного спуску:

1. Ініціалізація вагів:Встановлюються початкові значення вагів.

2. Пряме поширення (Forward Pass):Для кожних вхідних даних виконується проходження через нейронну мережу, щоб отримати прогнозоване значення.

3. Обчислення втрат:Обчислюється різниця між прогнозованим та фактичним значенням за допомогою функції втрат.

4. Зворотне поширення (BackwardPass):Обчислюються градієнти функції втрат щодо кожного параметра ваги та зміщення.Оновлюються ваги за допомогою методу градієнтного спуску.

5. Повторюйте кроки 2-4 для всіх вхідних даних, доки функція втрат не буде мінімізована до прийнятного рівня або не буде досягнута збіжність.

2.4 LeakyReLU

У попередніх частинах розглядалися функції активації [13] та втрат і їх важливість для нейронних мереж. Розберемо функції, які зазвичай використовуються з багатошаровими перцептронами (MLP).

Гіперболічна тангенсова функція (Tanh), що була використана в моделях 3-9-1 та 6-6-1,має матиматичну формулу:

$$\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)} \quad (2.1)$$

Це функція активації з S-подібною (сигмоїдною) формою графіка, що перетворює значення елемента на інтервал між -1 і 1.Однак у всіхсигмоїдних функцій активації є недолік, який називається проблемою затухання градієнта (vanishinggradientproblem). Це означає, що при використанні сигмоїдної функції активації в глибоких нейронних мережах градієнти можуть ставати дуже маленькими, що ускладнює навчання. У таких випадках часто використовується інша функція активації, наприклад, ReLU (Rectified Linear Unit).

ReLU [16] (Rectified Linear Unit) перетворює вхідне значення на значення від 0 до позитивної нескінченності. Якщо вхідне значення менше або дорівнює нулю, то функція активації видає нуль, в іншому випадку - вхідне значення.Математична формула визначається наступним чином:

$$relu(x) = \max(0, x) \qquad (2.2)$$

ReLU має декілька переваг порівняно із сигмоїдними функціями активації. По-перше, більша обчислювальна ефективність, оскільки вона є простою і швидкою операцією, яка не потребує обчислення експоненти. По-друге, ReLU вирішує проблему затухання градієнта, оскільки вона не викликає затухання градієнта під час зворотного поширення помилки, як це відбувається у випадку із сигмоїдною функцією активації.

Однак, є деякі недоліки, такі як “мертві” нейрони (dead neurons) і проблема “розшарування” (clustering). Для вирішення цих проблем можуть бути використані інші функції активації, такі як Leaky ReLU або ELU.

На відміну від ReLU, функція LeakyReLU повертає те саме значення, якщо input(x) позитивне, і повертає негативний ухил, якщо негативне. Негативним ухилом може бути будь-яке маленьке позитивно число, наприклад 0,01. Математична формула виглядає наступним чином:

$$elu(x) = \max(\alpha * x, x), \qquad (2.3)$$

де α— це негативний ухил.

Що ж стосується Swish [14], який використовується в моделі GraphCast, він має кілька ключових відмінних властивостей, які роблять його відміннимі кращим від ReLU та його модифікацій. По-перше, Swish є гладкою неперервною функцією, на відміну від ReLU, яка є кусково-лінійною функцією. Swish дозволяє поширювати невелику кількість від’ємних ваг, тоді як ReLU обнулює всі від’ємні вагові коефіцієнти. Проте, через підвищену

обчислювальну складність, ReLU виглядає більш економічно ефективним та простим у реалізації.

2.5 Методи регуляризації танормалізації

Вибір методу, будь то нормалізація або регуляризація, часто є складним завданням, оскільки найкращий підхід визначається на практиці та під час тестування. Водночас слід враховувати досвід, закладений у раніше розглянутих моделях, адже рішення їх авторів можуть слугувати корисним орієнтиром.Розглянемо основні методи регуляризації [15] та нормалізації [16]:

2.5.1 Регуляризація

Під час навчання в глибокій нейронній мережі існує ймовірність того, що мережа перестане підлаштовуватися під навчальні дані. Це відбувається тому, що коли багато нейронів у шарі отримують однакову інформацію з вхідних даних, це створює взаємозалежність між нейронами. Щоб запобігти надмірній підгонці, існує регулярізація даних.

Існує кілька типів методів регуляризації, які зазвичай використовуються в машинному навчанні, зокрема регулярізація L1 (Lasso), регулярізація L2 (Ridge) і регулярізація відсіву (Dropout).

L1 регуляризація, або Lasso, допомагає моделі вибирати важливі ознаки (фактори), ігноруючи незначні. Вона робить це, додаючи “штраф” за використання занадто багатьох ознак, що змушує деякі коефіцієнти стати нульовими. Це означає, що модель автоматично виключає неважливі ознаки. Перевага такого підходу в тому, що модель стає простішою та легшою для розуміння. Однак, якщо два ознаки сильно пов'язані між собою, L1 регуляризація може вибрати одну з них і відкинути іншу без конкретної причини.

L2 регуляризація, або Ridge, робить модель більш стабільною, зменшуючи вагу менш важливих ознак замість того, щоб їх повністю виключати. Вона додає “штраф” за великі коефіцієнти, тому ваги ознак зменшуються рівномірно. Це дозволяє всім ознакам залишатися в моделі, але їх вплив врівноважується. Перевага в тому, що модель стає стійкішою та більш рівномірною, особливо коли всі ознаки важливі. L2 регуляризація корисна, коли ми хочемо зберегти всі ознаки, але зменшити їх вплив, якщо вони не є дуже важливими. Також вона краще працює, коли ознаки сильно корелюють між собою, зменшуючи їх разом, а не вибираючи одну випадково.

Dropout [16] — це техніка регуляризації для нейронних мереж. Під час навчання випадковим чином “вимикають” деякі нейрони, встановлюючи їхні вихідні значення в нуль. Це запобігає надмірній залежності нейронів один від одного і робить мережу більш стійкою. Під час тестування Dropout не використовується, а вихідні значення масштабуються, щоб відповідати очікуваним значенням під час навчання. Таким чином, Dropout допомагає моделям не перенавчатися і ставати більш надійними.

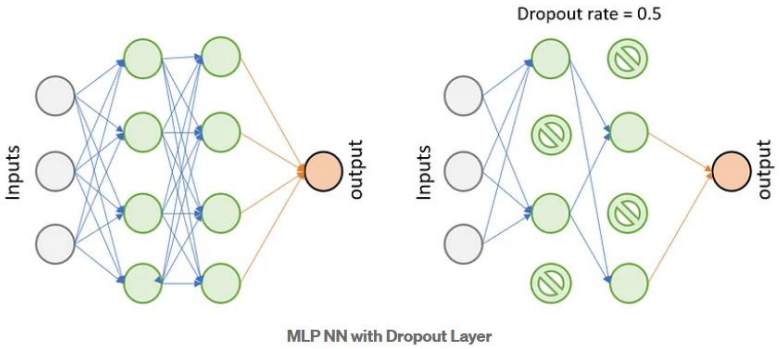

Рисунок 2.7 –MLPз шаром Dropout

2.5.2 Нормалізація

Нормалізація — це техніка попередньої обробки даних, яка використовується для перетворення значень числових стовпців у наборі даних у загальну шкалу без спотворення відмінностей у діапазонах значень або втрати інформації. Йдеться про налаштування масштабу ваших даних, щоб вирівняти умови гри для всіх функцій у вашому наборі даних.

Існує кілька поширених методів нормалізації, кожен із яких має своє застосування та переваги:

Мін-макс нормалізація (Min-MaxScaling) приводить дані до заданого діапазону, зазвичай [0,1][0, 1][0,1]. Формула виглядає так:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \qquad (2.4)$$

де x– початкове значення,x_{min} таx_{max} – мінімальне та максимальне значення у вибірці. Цей метод зберігає співвідношення між значеннями, але є чутливим до викидів.

Стандартизація(Z-ScoreNormalization, Standardization) приводить дані до нульового середнього та одиничного стандартного відхилення. Математична формула:

$$x' = \frac{x - \mu}{\sigma}, \qquad (2.5)$$

де μ – середнє значення вибірки, σ – стандартне відхилення. Вона ефективна для даних із нормальною розподіленістю, але чутлива до викидів.

Пакетна нормалізація (Batch Normalization, BatchNorm) виконується під час навчання, нормалізуючи активації нейронів у межах мініпакета. Це дозволяє прискорити навчання та використовувати вищі значення швидкості

навчання, але може бути неефективним на невеликих мініпакетах або в задачах з нерівномірними даними. Математична формула:

$$x' = \frac{x - \mu_{batc\ h}}{\sqrt{\sigma^2_{batc\ h} + \epsilon}}, \quad (2.6)$$

де $\mu_{batc\ h}$ – середнє значення в мініпакеті, $\sigma^2_{batc\ h}$ – дисперсія, ϵ – невелике число для уникнення ділення на нуль.

Шарова нормалізація (LayerNormalization, LayerNorm)нормалізує активації нейронів у межах одного зразка по всіх нейронах шару. Вона підходить для задач із послідовними даними та є незалежною від розміру мініпакета, хоча її ефективність для обробки зображень може бути нижчою. Саме ця нормалізація використовується в моделі GraphCast. Математична формула:

$$x' = \frac{x - \mu_{batc\ h}}{\sqrt{\sigma^2_{batc\ h} + \epsilon}}, \quad (2.7)$$

де $\mu_{batc\ h}$ та $\sigma^2_{batc\ h}$обчислюються для всіх нейронів одного шару. Це корисно для задач із послідовними даними.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОДЕЛІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Налаштування середовища розробки PyCharm

PyCharm — кросплатформенне інтегроване середовище розробки для мови програмування Python [17], розроблене компанією JetBrains на основі IntelliJIDEA.Він надає всі необхідні інструменти для зручної розробки, налагодження та тестування моделей машинного навчання. У цій частині розглядаються основні можливості та налаштування PyCharm для ефективної роботи.

Як вже було сказано, IDE працює з мовою програмування Python. Тому, для розробки, потрібно спочатку встановити її: інструкцію можна знайти на офіційному сайті (version==3.11).

Для встановлення Py Charm потрібно скористуватись сайтом JetBrains [18]. Існує дві версії: Community Edition (безкоштовна) та Professional Edition (комерційна, з розширеними функціями). Для більшості задач Community Edition є достатньою.

Після встановлення PyCharm, потрібно створити новий проект або відкрити існуючий, вибрати встановлену версію інтерпретатора Python, який буде використовуватись у проекті. Рекомендується також використовувати технологію “.venv”. Технологія “.venv” допомагає створити віртуальне середовище для вашого проекту, що забезпечує більш чисту систему та запобігає конфліктам залежностей. але можна обійтись і без неї.

Лістинг 3.1–Команда створення .venv-середовища

```
python -mvenvvenv
```

Лістинг 3.2–Активація віртуального середовища на Windows

```
venv\Scripts\activate
```

Лістинг 3.3–Активація віртуального середовища на macOS/Linux

```
sourcevenv/bin/activate
```

Для встановлення сторонніх бібліотек використовується інсталятор PIP – один із найпоширеніших пакетних менеджерів для мови програмування Python.

Список бібліотек, які потрібно встановити, прописується в файлі “requirements.txt”. Для встановлення використовується команда:

Лістинг 3.4– Встановлення бібліотек

```
pipinstall -rrequirements.txt
```

Лістинг 3.5– Список бібліотек, потрібних для проекту

```
cdsapi>=0.7.2
netCDF4
scikit-learn
tensorflow[and-cuda]  #for not windows use tensorflow
cloud-tpu-client
matplotlib
```

3.2 Завантаження та попередня обробка даних

3.2.1 Отримання даних по API

Як зазначалось в попередніх частинах, найкращим варіантом для навчання моделей для прогнозування погоди є ERA5від ECMWF [19].

Потрібно пройти реєстрацію на сайт та заповнити профіль. У розділі“Datasets”треба знайтиз назвою “ERA5 hourlydataonsinglelevelsfrom 1940 topresent”.Післянатискання на кнопку “Download” потрібно знайти кнопку“Termsofuse” та натиснути кнопку “Accept”.

Після цього можна використовувати APIсайта [20].

Лістинг 3.6 – Код файлу “cds_climate_api.py”

```
import cdsapi

from const import NC_FILE_NAME


def get_cds_climate_dataset():
    dataset = "reanalysis-era5-single-levels"
    request = {
        "product_type": ["reanalysis"],
        "variable": ["2m_temperature"],
        "year": [str(year) for year in range(1990, 2025)],
        "month": [f"{month:02d}" for month in range(1, 13)],
        "day": [f"{day:02d}" for day in range(1, 32)],
        "time": ["00:00", "06:00", "12:00", "18:00"],
        "data_format": "netcdf",
        "download_format": "unarchived",
        "area": [50.1, 36, 49.9, 36.2]
    }
```

```
    client = cdsapi.Client()
    client.retrieve(dataset, request,
target=NC_FILE_NAME).download()# NC_FILE_NAME=”climate.nc”
```

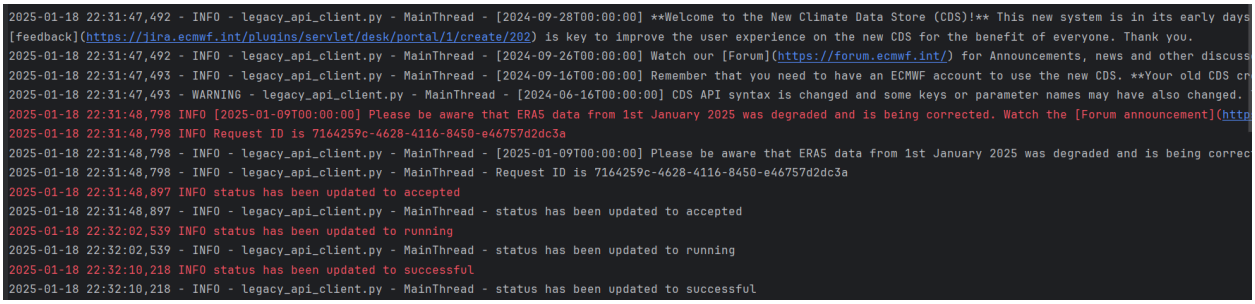

Рисунок 3.1 – Логи cdsapi.Client()

Після виконання в директорії проекту з’явиться“climate.nc” файл.

3.2.2 Попередня обробка даних

Отриманий “climate.nc” потрібно попереднє обробити, щоб використати дані для навчання моделі.

Лістинг 3.7 – Код файлу “netcdf.py”

```
import logging

import numpy as np
import netCDF4

from const import NC_FILE_NAME


def convert_nc_to_array():
    dataset = netCDF4.Dataset(NC_FILE_NAME)
    logging.debug(dataset.variables)
```

```
    # Open the netCDF file
t2m_var = dataset.variables["t2m"]
time_var = dataset.variables["valid_time"]

    # Convert temperature from Kelvin to degrees Celsius
t2m_celsius_arr = t2m_var[:] - 273.15

# Reshape arrays
    times_x_arr = time_var[:].reshape(-1, 1)
t2m_y_arr = t2m_celsius_arr.flatten()

min_time, max_time = times_x_arr.min(),times_x_arr.max()
    norm_times_x_arr = (times_x_arr - min_time) / (max_time
    - min_time)
          times_x_arr=np.column_stack((t2m_y_arr,norm_times_x
     _arr))

    return times_x_arr, t2m_y_arr
```

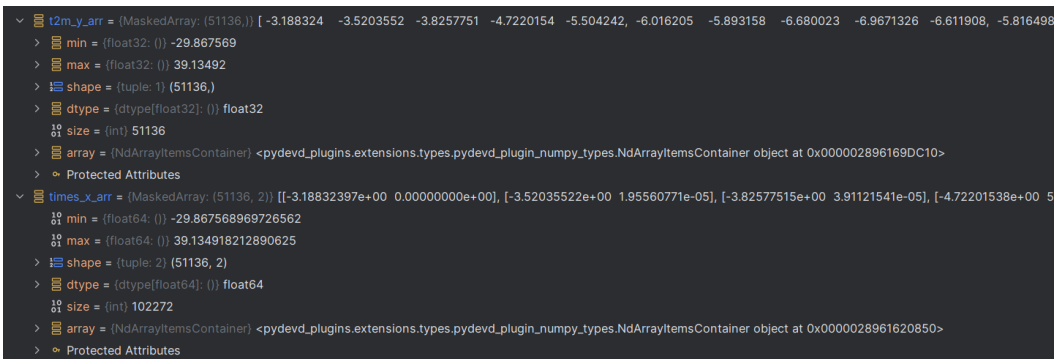

Рисунок 3.2–“t2m_celsius_arr” та “times_arr”у режимі відладки

3.3Оголошення моделіMLP

MLP намагається вивчити шаблон або відобразити вхідні та вихідні дані за допомогою вагових коефіцієнтів, навчаючись із заданими даними. Отже, щоб досягти цього, використовується розглянутий раніше алгоритм зворотного поширення помилки та стохастичний градієнтний спуск. Втрати

обчислюються за допомогою RMSE. Для реалізації використовується TensorFlow [23].

TensorFlow — один із найпотужніших і найпопулярніших фреймворків для машинного навчання, розроблений компанією GoogleBrain у 2015 році. Бібліотека підтримує повний цикл розробки та застосування моделей машинного навчання — від підготовки даних і навчання до розгортання та оптимізації. Фреймворк пропонує кілька рішень для розгортання моделей, включаючи TensorFlowServing для хмарного середовища, TensorFlowLite для мобільних пристроїв та IoT, а також TensorFlow.js для роботи у браузері.

Архітектура складається з одного вхідного, трьох прихованих і одного вихідного шару. Два вхідних нейрони: температура та час. Кількість нейронів у шарах зменшується наступним чином: 128-64-32-16-1. На вхідному шарі використовується LayerNormalization. У прихованих шарах застосовується функція активації LeakyReLU, а також в одному з шарів використовується пакетна нормалізація і регулярізація Dropout з коефіцієнтом 0,2.

Лістинг 3.8 – Код файлу “mlp.py”

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.optimizers import SGD
import tensorflow.keras.backend as K

# Define RMSE loss function
def rmse(y_true, y_pred):
    return K.sqrt(K.mean(K.square(y_pred - y_true)))

# Build the model with several hidden layers and
normalization
model = Sequential([
    layers.Dense(128, input_shape=(2,)),
    layers.LayerNormalization(),
```

```
    layers.Dense(64),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.01),
    layers.Dropout(0.2),

    layers.Dense(32),
    layers.LeakyReLU(alpha=0.01),

    layers.Dense(16),
    layers.LeakyReLU(alpha=0.01),

    layers.Dense(1)
])

# Compile the model using SGD and RMSE
sgd = SGD(learning_rate=0.001, momentum=0.9)
model.compile(optimizer=sgd, loss=rmse, metrics=["mae"])
```

3.4Навчання нейронної мережі

Окрім навчання, потрібні ще дані для валідації. Отримані дані (лістинг 3.7) поділяються на 2 пакета, 70% - на навчання. 30% - на валідацію. Отриманий маленький пакет використовується для перевірки кінцевого значення RMSE. Кінцеве значення RMSE~ 0.795.

Лістинг 3.9 –Навчання та оцінювання моделі

```
# Convert netCDF to arrays
times_x_arr, t2m_y_arr = convert_nc_to_array()

from sklearn.model_selection import train_test_split

# Split the data into training and test sets
```

```
times_x_arr_train,     times_x_arr_test,     t2m_y_arr_train,
t2m_y_arr_test   =   train_test_split(times_x_arr,   t2m_y_arr,
test_size=0.3, random_state=42)

# Train the model with changes
history = model.fit(times_x_arr_train, t2m_y_arr_train,
epochs=100,batch_size=32, validation_split=0.2, verbose=1)

# Evaluate the model and accuracy
rmse, mae = model.evaluate(times_x_arr_test,
t2m_y_arr_test, batch_size=32, verbose=1)
logging.info("RMSE: {0}".format(rmse))
logging.info("MAE: {0}".format(mae))
```

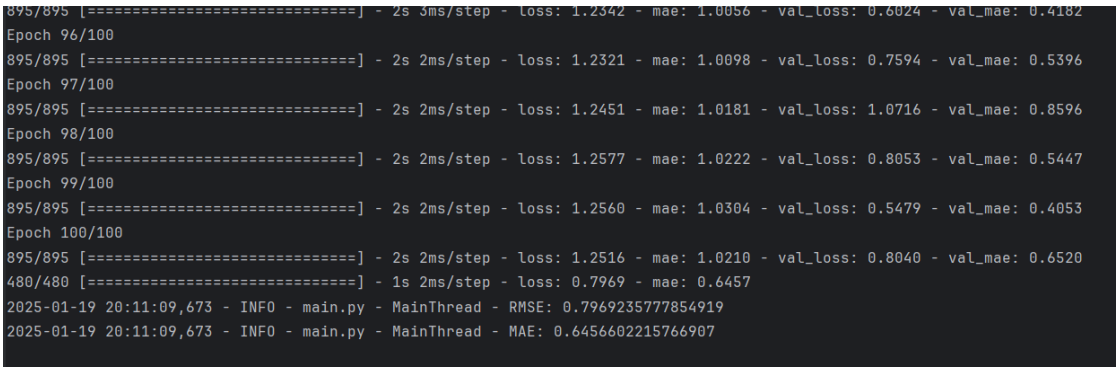

Рисунок 3.3– Логі виконання програми після навчання моделі

3.5 Аналіз результатів

Для створення зображень та візуалізації даних використовується “matplotlib”—бібліотека для візуалізації даних за допомогою двовимірної та тривимірної графіки.

Лістинг 3.10– Код файлу “plot.py”

```
import matplotlib.pyplot as plt
```

```
def show_rmse_and_mae_plots(history):
    # Visualize training history
    plt.figure(figsize=(12, 4))

    # Plot loss
    plt.subplot(1, 2, 1)
    plt.plot(history.history["loss"], label="Train Loss")
    plt.plot(history.history["val_loss"], label="Test Loss")
    plt.xlabel("Epochs")
    plt.ylabel("Loss (RMSE)")
    plt.legend()
    plt.title("Training and Testing Loss")

    # Plot MAE
    plt.subplot(1, 2, 2)
    plt.plot(history.history["mae"], label="Train MAE")
    plt.plot(history.history["val_mae"], label="Test MAE")
    plt.xlabel("Epochs")
    plt.ylabel("MAE")
    plt.legend()
    plt.title("Training and Testing MAE")

    plt.show()
```

Графіки RMSE та MAE [22](рисунок 3.4) є важливими інструментами для оцінки якості прогнозів моделі. RMSE показує середню квадратичну помилку між фактичними значеннями та прогнозами моделі, що дозволяє краще розуміти її продуктивність. Чим менше значення RMSE, тим точнішими є прогнози моделі. MAE показує середню абсолютну помилку між фактичними значеннями та прогнозами моделі, яка вимірює середню різницю між значеннями. Це допомагає розробникам і аналітикам оцінювати та вдосконалювати свої моделі машинного навчання для досягнення кращих результатів.

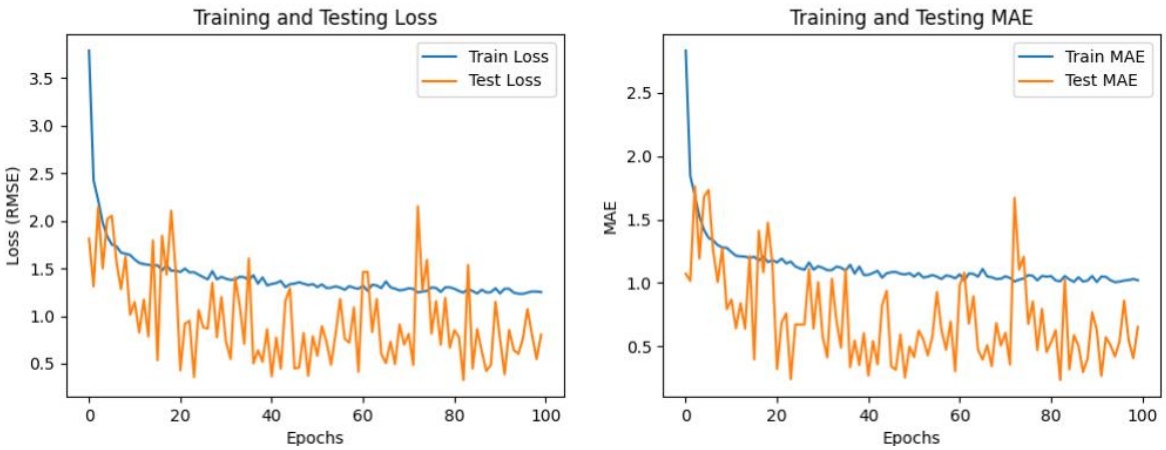

Рисунок 3.4 – ГрафікRMSE та MAE

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АПАРАТНОЇ ЧАСТИНИ

4.1 ArduinoNano 33 BLESense

ArduinoNano 33 BLESense [6]— це плата розробника від Arduino з підтримкою штучного інтелекту у форм-факторі практично ArduinoNANO з набором датчиків, які дозволять вам без будь-якого зовнішнього обладнання відразу ж пристати до розробки IoT-пристроїв на базі AI.

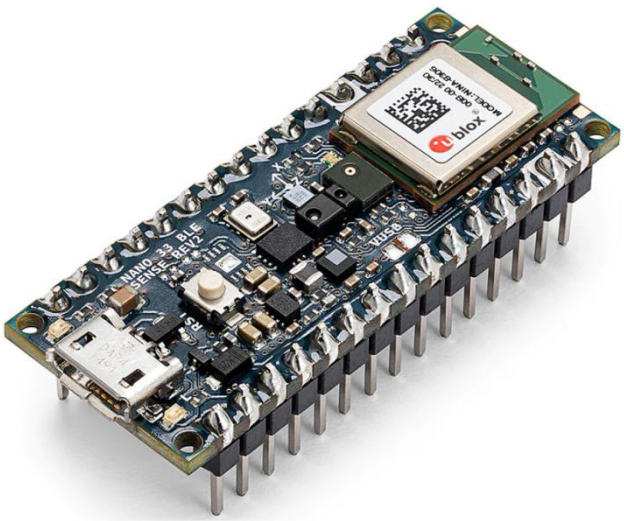

Рисунок 4.1 –Arduino Nano 33 BLE Sense Rev2 ABX00070

Периферійні пристроїв становлені на платі:

- IMU: BMI270 і BMM150.
- Мікрофон: MP34DT06JTR.
- Датчик навколишнього середовища (освітленість, наближення, виявлення/розпізнавання жестів і кольору): APDS9960.
- Датчик тиску: LPS22HB

- Датчик температури і вологості: HS3003

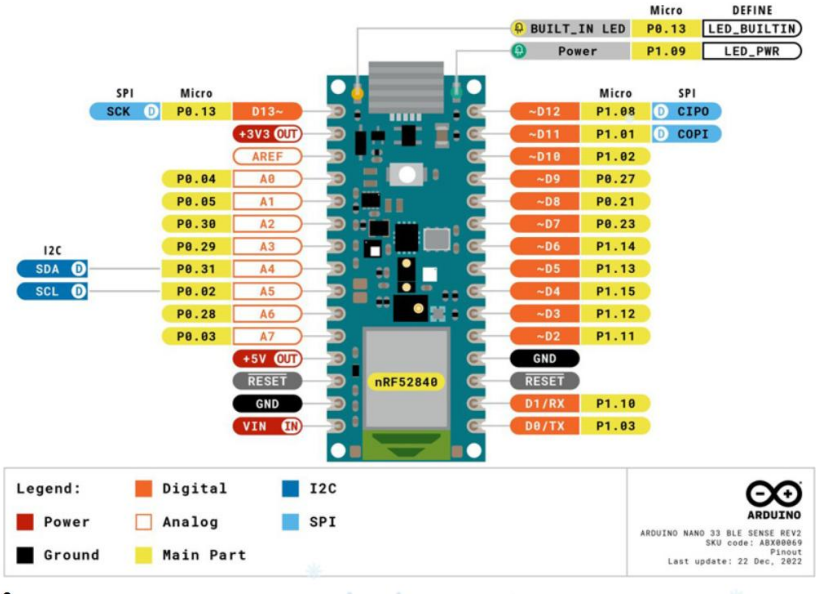

Рисунок 4.2– Схема Arduino Nano 33 BLE

Головною особливістю цієї плати є можливість запуску на ній застосувань з підтримкою AI і моделі Edge Computing з використанням TinyML.

Edge Computing - це модель обробки даних, при якій обробка і зберігання даних відбувається максимально близько до джерела даних або безпосередньо на самому пристрої. Використовуючи цю модель, відпадає необхідність у централізованих серверах і хмарних сховищах.

Користувач також може використовувати TensorFlow Lite і Edge Impulse.

TensorFlow Lite (TFLite) - це вільна та відкрита бібліотека, призначена для виконання моделей машинного навчання на пристроях з обмеженими ресурсами.

4.2 Конвертація навчанної моделі

Для того, щоб використати модель, написану на tensorflow (лістинг 3.8), її потрібно конвертувати.

Лістинг 4.1 – Конвертація моделі

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

#Create a file containing our tflite model
open("tflite_model.tflite","wb").write(tflite_model)
```

Отриманий файл потрібно перевести із бінарного виду у текстовий формат. Для цієї мети підходить утиліта “xxd”.

Лістинг 4.2 – Установка xxd на Windows

```
chocoinstallxxd
```

Лістинг 4.3– Установка xxdна Linux

```
apt-getinstall -qqxxd
```

“tflite_model.tflite” перетворюється у шістнадцяткове представлення та додається до файлу “model.h”

Лістинг 4.4– Генерація файлу model.h

```
cat ./tflite_model.tflite | xxd -i>>model.h
```

Вміст файлу потрібно зберегти у змінну “constunsignedcharmodel[]”.

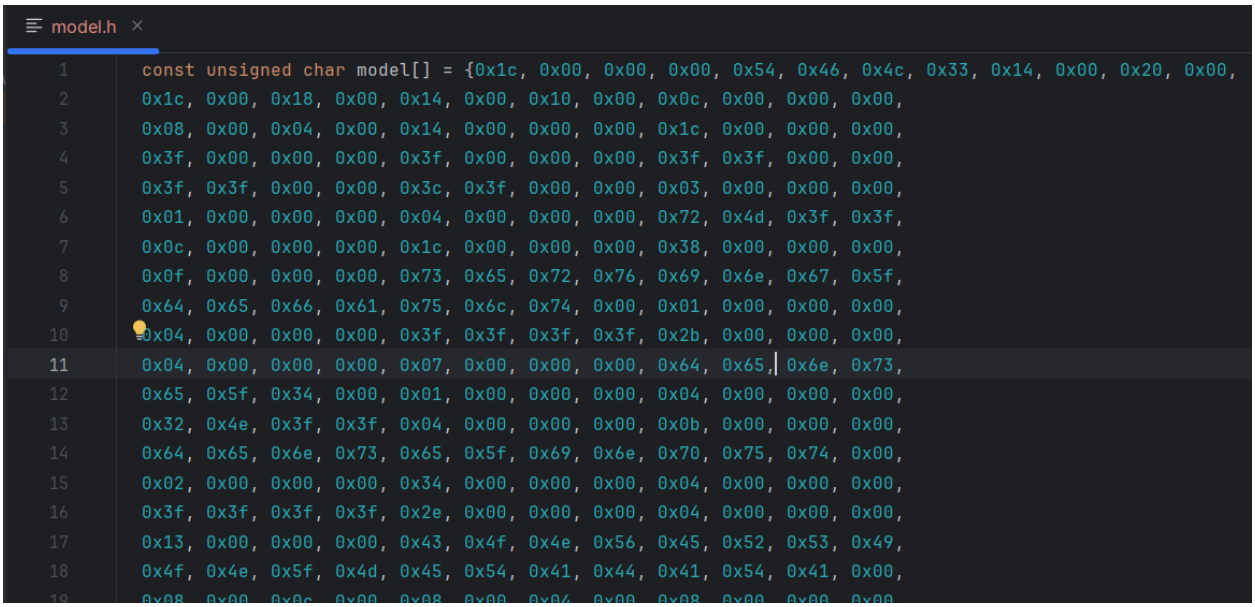

Рисунок 4.3– Файл “model.h”

4.3Інтеграція моделі з реальними даними

Для написання cpp-коду використовується VisualStudioCode [21].

VisualStudioCode (VSCode) — текстовий редактор, розроблений Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформової розробки веб- та хмарних застосунків. Включає в себе відладчик, інструменти для роботи з Git, підсвітку синтаксису, IntelliSense та засоби для рефакторінгу.

Для того, щоб встановити сторонні embedded бібліотеки використовується PlatformIOIDE. Встановлюється IDE через розділ “Extensions” у VisualStudioCode.

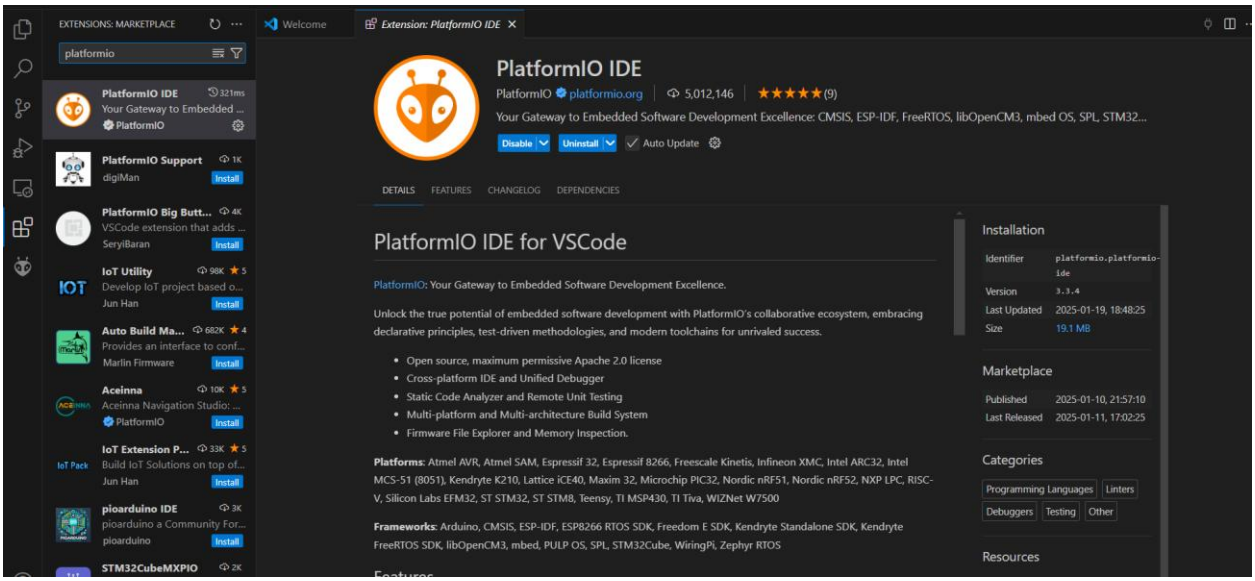

Рисунок 4.4 – Інсталяція PlatformIOIDE

У директорії Arduino створюється вручну файл “platformio.ini”. Без цього файлу PlatformIOIDE не запрацює.

Лістинг 4.5 – Зміст файлу “platformio.ini”

```
[env:nano33ble]
platform = nordicnrf52
board = nano33ble
framework = arduino
monitor_speed = 9600
lib_deps =
   eloquentarduino/EloquentTinyML@^0.0.3
   arduino-libraries/Arduino_HTS221@^1.0.0
   arduino-libraries/RTCZero@1.6.0
```

Тепер IDEможе ініціалізуватись та відкриватись. В розділі “Libraries” кнопкою “AddtoProject” додаються бібліотеки:

1. “Arduino_HTS221” для використання температурного сенсору плати.

2. “EloquentTinyML” для інтегрування моделі TensorFlowLite (лістинг 4.1) в проект.

3. “RTCZero” для отримання часу в форматі unix.

Виконується установка бібліотек натисканням на кнопку “Addto Project” у розділі “Libraries”PlatformIO.

Скомпільований файл “model.h” переноситься в директорію “include”.

Лістинг 4.6 – Зміст файлу “main.cpp”

```
#include <Arduino.h>
#include "model.h"
#include <EloquentTinyML.h>
// Library for the temperature sensor of the NANO 33 BLE Sense
#include <Arduino_HTS221.h>
// Library for the real-time clock
#include <RTCZero.h>

// Number of inputs
#define INPUTS 1
// Number of outputs
#define OUTPUTS 1
// Memory you will allocate to the model
#define Tensor_ARENA 2*1024

// Declare our model
Eloquent::TinyML::TfLite<INPUTS, OUTPUTS, Tensor_ARENA> weather_model;

// RTC for time tracking
```

```
RTCZero rtc;

void setup()
{
  // Start the serial communication
  Serial.begin(9600);

  // Initialize RTC
  rtc.begin();

  // Start the model
  weather_model.begin(model);

  // Instantiate the temperature sensor
  if (!HTS.begin())
  {
    Serial.println("Failed to initialize temperature sensor!");
    while (1)
      ;
  }
}

void loop()
{
  // Read temperature in °C
  float temp = HTS.readTemperature();

  // Create an array containing unix-time
  float inputs[INPUTS] = {rtc.getEpoch()};
  // Create an empty array that will contain the result of the prediction
  float outputs[OUTPUTS] = {0};
  // Predict the inputs and fill the outputs arguments with the neural network output
```

```
  weather_model.predict(inputs, outputs);

  Serial.print("Real temperature : ");
  Serial.print(temp);
  Serial.print("Predicted temperature : ");
  Serial.println(outputs[0]);
  delay(60000);
}
```

Написаний код отримає дані з сенсора контролера та кожну хвилину виводить реальну температуру та температуру спрогнозовану моделлю.

```
Real temperature : 23.45 Predicted temperature : 22.66
Real temperature : 23.50 Predicted temperature : 22.71
Real temperature : 23.55 Predicted temperature : 22.76
Real temperature : 23.60 Predicted temperature : 22.81
Real temperature : 23.65 Predicted temperature : 22.86
```

Рисунок 4.5–Логі виконання програми main.cpp

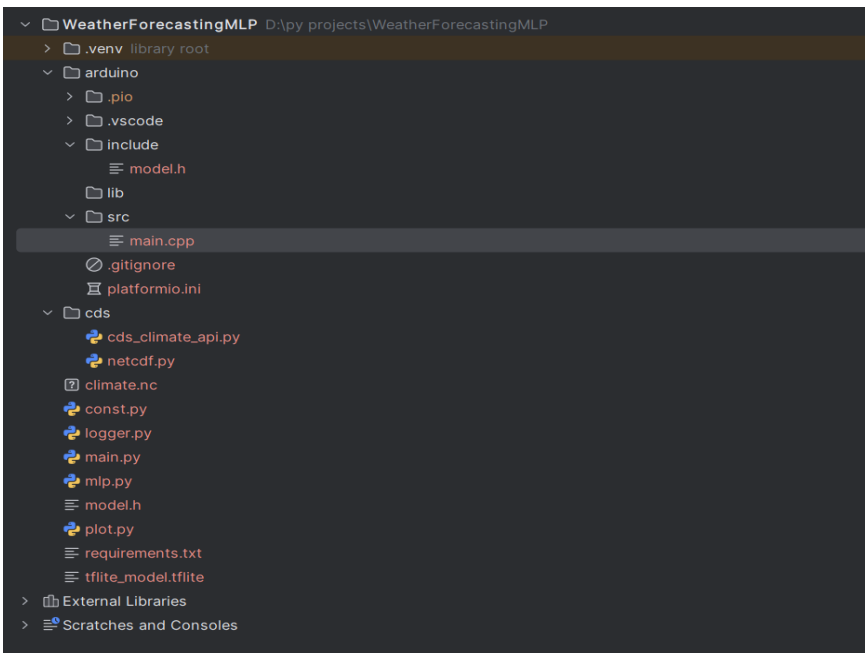

Рисунок 4.6– Структура проекту з прогнозування погоди

ВИСНОВКИ

У ході виконання роботи були успішно досліджені можливості використання технологій штучного інтелекту для задач прогнозування погоди. Було вивчено теоретичні основи штучного інтелекту та метеорології. Розглянуто та проаналізовано сучасні наукові праці та реалізації в області прогнозування погоди, особливості яких згодом були використані на практиці. Окрему увагу у другому розділі роботи було приділено моделі 3-9-1, моделі 6-6-1 та моделі GraphCast.

У ході дослідження було встановлено, що багатошаровий перцептрон (Multi-Layered Perceptron) є найбільш відповідним варіантом для задач прогнозування погоди. Серед методів навчання було обрано алгоритм зворотного поширення помилки у поєднанні з алгоритмом оптимізації градієнтного спуску, описано схему навчання.

Проаналізовано методи активації, виділено їх сильні та слабкі сторони, результатом чого став вибір функції Leaky ReLU. Було розглянуто техніки нормалізації та регуляризації, описано їх особливості.

У процесі розробки моделі було описано метод отримання даних ERA5 через API у форматі *.nc. Дані були оброблені, приведені до необхідного для навчання вигляду, після чого використані для навчання та валідації моделі, побудованої на фреймворку TensorFlow [23]. Також було побудовано графіки RMSE та MAE для аналізу результатів навчання.

Під час розробки програмного забезпечення для апаратної частини було розглянуто плату Arduino Nano 33 BLE Sense, виділено її функціональні можливості. Написана мовою Python модель була конвертована за допомогою TensorFlow Lite у файл “model.h”, після чого раніше навчена нейронна мережа змогла працювати у зв’язці з реальними даними.

Продемонстрована архітектура нейронної мережі показала хороші результати навіть на відносно невеликому обсязі даних для навчання. При

цьому підбір гіперпараметрів моделі не був детально оптимізований, що свідчить про значний потенціал для подальшого вдосконалення архітектури. З апаратної точки зору, описаний підхід до інтеграції навченої моделі демонструє можливість використання подібних систем для створення більш складних комплексів. Наприклад, на основі цієї моделі можна розробити інтегровані метеорологічні станції, що об’єднують аналіз температури, вологості, тиску та інших параметрів, забезпечуючи не лише точне прогнозування погоди, але й виявлення екстремальних кліматичних явищ у реальному часі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Cloud Google. What is Artificial Intelligence? [Електронний ресурс] – Режим доступу: https://cloud.google.com/learn/what-is-artificial-intelligence#what-is-artificial-intelligence-ai
2. Wikipedia. Machinelearning [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Machine_learning
3. Wikipedia. Deeplearning [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Deep_learning
4. Coursera. Neural Network Architecture [Електронний ресурс] –Режим доступу: https://www.coursera.org/articles/neural-network-architecture
5. Wikipedia. Meteorology [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Meteorology
6. Arduino. ArduinoNano 33 BLESenseRev2 withheaders [Електронний ресурс] – Режим доступу: https://arduino.ua/ru/prod6293-arduino-nano-33-ble-sense-rev2-with-headers
7. Lam, R., Sanchez-Gonzalez, A., Willson, M., et al. Learning skillful medium-range global weather forecasting. Science, 14 Nov 2023, Vol. 382, Issue 6677, pp. 1416-1421
8. Fahimi Nezhad, E., Fallah Ghalhari, G., & Bayatani, F. (2019). Forecasting maximum seasonal temperature using artificial neural networks “Tehran case study”. Asia-Pacific Journal of Atmospheric Sciences, 55, 145-153
9. Madhiarasan, M., Tipaldi, M., & Siano, P. (2020). Analysis of artificial neural network performance based on influencing factors for temperature forecasting applications. Journal of High-Speed Networks, 26(3), 209-223
10. LoginomWiki. MultilayeredPerceptron [Електронний ресурс] – Режим доступу: https://wiki.loginom.ru/articles/multilayered-perceptron.html

11. BackpropagationinNeuralNetwork [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/backpropagation-in-neural-network
12. Neurohive. GradientDescent [Електронний ресурс]. –Режим доступу: https://neurohive.io/ru/osnovy-data-science/gradient-descent
13. Habr. ChoosingActivationFunction [Електронний ресурс]. – Режим доступу – Режим доступу: https://habr.com/ru/articles/727506
14. DigitalOcean. SwishActivationFunction [Електронний ресурс]. – Режим доступу: https://www.digitalocean.com/community/tutorials/swish-activation-function
15. Microsoft. Test Run L1 and L2 Regularization for Machine Learning [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/ru-ru/archive/msdn-magazine/2015/february/test-run-l1-and-l2-regularization-for-machine-learning
16. Elfwing, S.; Uchibea, E.; Doya, K. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning / S. Elfwing, E. Uchibea, K. Doya. – 2020
17. Python. OfficialDocumentation [Електронний ресурс] –Режим доступу: https://www.python.org/downloads
18. JetBrains. PyCharm. [Електронний ресурс] –Режим доступу: https://www.jetbrains.com/pycharm
19. CopernicusClimateDataStore. [Електронний ресурс] –Режим доступу: https://cds.climate.copernicus.eu
20. Copernicus Climate Data Store API. [Електронний ресурс] –Режим доступу: https://cds.climate.copernicus.eu/how-to-api
21. CodeVisualStudio. [Електронний ресурс] — Режим доступу: https://code.visualstudio.com

22. Towards Data Science. What are RMSE and MAE? [Електронний ресурс] — Режим доступу: https://towardsdatascience.com/what-are-rmse-and-mae-e405ce230383

23. TensorFlow. [Електронний ресурс] — Accessed: https://www.tensorflow.org