Нейронна мережа

**НЕЙРОННА МЕРЕЖА З ПОШКОДЖЕННИМИ ЗВ’ЯЗКАМИ**

Наукова робота

на Всеукраїнський конкурс студентських наукових робіт з природничих, технічних та гуманітарних наук у 2020/2021 навчальному році

зі спеціальності «Електроніка»

**ЗМІСТ**

[**1.** **ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ У СИСТЕМАХ SMARTGRID** 4](#_Toc63327956)

[**1.1 Нейронні мережі в інтелектуальних системах** 4](#_Toc63327957)

[**1.2.** **Властивості та структура нейронних мереж** 5](#_Toc63327958)

[**1.3.** **Рекурентні нейронні мережі (RNN)** 6](#_Toc63327959)

[**1.4.** **Вибір моделі** 7](#_Toc63327960)

[**1.5.** **Пошкодження у нейронних мережах** 8](#_Toc63327961)

[**РОЗДІЛ 2. РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ** 10](#_Toc63327962)

[**2.1.** **Опис роботи нейронної мережі** 10](#_Toc63327963)

[**3.2.** **Пошкодження другого шару мережі** 15](#_Toc63327964)

[**ВИСНОВКИ** 22](#_Toc63327965)

[**ПЕРЕЛІК ВИКОРИСТАНОЇ НАУКОВО-ТЕХНІЧНОЇ ЛІТЕРАТУРИ** 23](#_Toc63327966)

[**ДОДАТОК А** 26](#_Toc63327967)

**ВСТУП**

Важливою сферою досліджень останні кілька років є прогнозування температури навколишнього середовища. Складність аналізу полягає в мінливому характері погодних даних, а оскільки дані про погоду хаотичні, то штучна нейронна мережа (ШНМ) стала однією з найкращих технік для виявлення залежностей між різними чинниками, що впливають на значення температури навколишнього середовища. У роботі розглядається застосування ШНМ для прогнозу температури навколишнього середовища, а також порівнюється точність розроблених моделей, які відрізняються між собою кількістю нейронів в шарах [1]. Передбачення про температуру є важливими прогнозами, тому що вони використовуються для захисту життя та власності.

Прогнозування температури - це застосування науки та техніки для прогнозування стану температури на майбутній час та певного місця. Прогнози температури робляться шляхом збору кількісних даних про поточний стан атмосфери. У цій роботі представлений алгоритм прогнозування температури на основі нейронної мережі

На рис.1.1 зображені професіональні сфери, в роботі яких, застосовуються штучні нейронні мережі. В медицині, ШНМ використовують для визначення діагнозу та прогнозу протікання процесів, необхідних для одужання пацієнта. Завдяки можливості якісно працювати с великим обсягом математичних даних, ШНМ застосовуються в економіці. Вони дають оцінку ризику неповернення кредитів, передбачення банкрутств, оцінку вартості нерухомості.

В промисловості, завдяки нейронним мережам, здійснюється оптимізація режимів виробничого процесу, комплексна діагностика якості продукції, моніторинг і візуалізація багатовимірної диспетчерської інформації, попередження аварійних ситуацій.



Рис.1.1. Галузі застосування штучних нейронних мереж

Через можливість аналізувати закономірності масових процесів, нейронні мережі здатні прогнозувати стан трафіку на дорогах та визначати швидший маршрут в логістиці. Аналізуючи рис 1.1, прогнозування — це задача, яка вирішуються майже в усіх галузях. Енергетика — це найбільша технологічна сфера, яка використовує ШНМ. Енергозбереження — один із актуальних напрямків застосування передбачення.

Прогнозування температури навколишнього середовища — це застосування технічних засобів, для передбачення майбутнього стану температури для певного регіону в заданий час. Прогнозування значень температури відбувається шляхом збору кількісних даних про поточний стан атмосфери, таких як: вологість повітря, атмосферний тиск, швидкість вітру, освітленість та видимість. Дані, зібрані з метеостанцій та супутників, які використовуються для аналізу та прогнозу температури, застосовуються для покращення ефективності використання енергетичних ресурсів. Завдяки передбаченню температури, можливий контроль систем кондиціювання та нагріву, що значно підвищує енергоефективність об’єкту.

На помилку прогнозу впливають декілька чинників, а саме: хаотичний характер зміни температурних значень та неточне вимірювання початкових даних, які використовуються для тренування нейронної мережі. Внаслідок чого, прогнози стають менш точними. З розвитком науки та технологій, можливо отримати крайнє точні прогнози значень температури незалежно від пори року або локації, для яких проводиться передбачення.

 В роботі, рекурентна нейронна мережа з довгою короткостроковою пам’яттю (LSTM) використовується, як програмна технологія для прогнозування та аналізу майбутнього значення температури навколишнього середовища на основі даних за 4,5 місяці.

1. **ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ У СИСТЕМАХ SMARTGRID**

**1.1 Нейронні мережі в інтелектуальних системах**

SmartGrid - це новітня всесвітня технологія, яка розглядає процес покращення технічних показників в сфері електроенергетики. Комплект цифрових інновацій для покращення енергетичної інфраструктури притаманний, як для споживачів, так і для постачальників електроенергії та послуг. Технологія започаткувалась в США на початку минулого століття, а на протязі останнього десятиліття впроваджується майже у всіх країнах світу.

Деякі фактори стали передумовою для появи SmartGrid, а саме: зменшення природних енергетичних ресурсів, екологічні зміни навколишнього середовища, збільшення потреб користувачів та кількості використовуваної електроенергії.

Історичним поштовхом для розвитку SmartGrid стало неякісне перебійне електропостачання в стратегічних містах США наприкінці минулого століття. Технологічною передумовою для модернізації енергетичної інфраструктури були наукові досягнення в області комп’ютерних технологій, електроніки та енергетики.

Для побудови аналізуючої та керуючої частини в розумній мережі SmartGrid використовуються штучні нейронні мережі (ШНМ). ШНМ — це математична модель, або ж її апаратне втілення, яка базується на принципах функціювання біологічної нейронної мережі в людському мозку.

Приклади задач, які нейронні мережі можуть виконувати в інтелектуальних мережах SmartGrid: 1) Керування системою чи окремими її секціями; 2) Розпізнавання режиму роботи; 3) Передбачення різного виду навантаження; 4) Аналіз даних для передбачення аварій; 5) Аналіз минулих аварійних станів для їх уникнення в майбутньому; 6) Регулювання певних показників (напруги, температури, тощо); 7) Виявлення помилок та несправностей;

Баланс виробництва і споживання електроенергії - це основа технологічної стійкості енергосистеми, його порушення позначається на якості електроенергії (відбувається деградація частоти і напруги в мережі), що знижує ефективність роботи обладнання. Крім того, правильний прогноз дозволяє забезпечити оптимальний розподіл навантаження між об'єктами енергосистеми [7].

* 1. **Властивості та структура нейронних мереж**

На рис. 1.2 представлена структура штучного нейрону, на якому $w\_{i}$представлено вектор вхідних сигналів, позначений як $x\_{i}$, що надходить на штучний нейрон. Вектор сигналів $x\_{i}$ множиться на вектор вагів, і надходить на блок-суматор, позначений Σ. Кожна вага відповідає «силі» однієї біологічної синаптичного зв'язку.



Рис.1.2 Структура штучного нейрону

Блок-суматор, що відповідає тілу біологічного елемента, складає зважені входи алгебраїчно, створюючи вихід, який має назву S. В свою чергу, отримане значення S служить аргументом для функції активації. Саме ця функція визначає значення вихідного сигналу нейрона, що потім надсилається на головний вихід нейрона y. Отже, штучні нейрони об'єднуються шляхом з'єднання виходів та входів між собою. Нейрон складається з трьох основних елементів:

1. ваги (множники), які характеризують важливість зв’язку між нейронами в системі;
2. суматор, що проводить операцію додавання вхідних сигналів, перед цим помножених на відповідні ваги;
3. функція активації, яка зв’язує вхід та вихід системи.

Теоретично описавши, можливо математично зобразити будову нейрона наступним чином:

** (1.1)

де, xi— вектор вхідних даних, wi — вага, bi — зсув, y — вихід нейрона.

* 1. **Рекурентні нейронні мережі (RNN)**

Рекурентна нейронна мережа — це модель глибокого навчання, яка використовується для прогнозування, розпізнавання, тощо. На відміну від традиційних нейронних мереж, рекурентні мережі використовують свою пам'ять (стани) для прогнозування результатів послідовностей, тобто, рекурентна нейронна мережа використовується у випадку передбачення майбутнього результату на основі попередніх послідовних вхідних сигналів. Ключовою особливістю цієї архітектури є обмін інформацією між кроками по часу.

На рис. 1.3 зображено модель рекурентної мережі, де h(t) – модифікований стан, h(t-1) – попередній стан, тоді як x(t) – сигнал поточних даних, A – комірка пам’яті, t = 0,1,2, …, n – часовий крок.



Рис.1.3. Модель рекурентної мережі в розгорнутому вигляді

Алгоритм роботи рекурентної нейронної мережі:

1. Х(t) подається в мережу.
2. Обчислюється поточний стан сигналу, використовуючи комбінацію поточного вводу та попереднього стану, тобто обчислюється h(t).
3. Поточний h(t) стає h(t-1) для наступного кроку (проходження часових кроків здійснюється в такому обсязі, якого цього вимагає проблема).
4. Здійснюється проходження всіх попередніх кроків для обчислення виходу.
5. Результат порівнюється з фактичним результатом і генерується помилка.
6. Помилка передається в мережу для оновлення ваг і мережа навчається.

У звичайних рекурентних мережах існують певні труднощі з вирішенням проблем, де довгострокові часові залежності є вирішальними. Причина полягає в тому, що під час зворотного розповсюдження, вплив попередніх прихованих шарів занепадає експоненціально з кожним часовим кроком.

Це явище зветься зникаючим градієнтом, і його неможливо подолати, просто збільшуючи вплив попередніх кроків часу, оскільки, в кінцевому підсумку, це призведе до вибуху градієнту, де вплив зв’язку зворотного зв’язку зростає сильніше, ніж вплив вхідного сигналу.

* 1. **Вибір моделі**

При виборі моделі, розглядались головні види ШНМ, ґрунтуючись на задачі прогнозування. Для прогнозування температури навколишнього середовища потрібна мережа, яка має змогу ідентифікувати непередбачуваний спад чи навпаки підйом температури.

Спершу розглядалися моделі ШНМ, які працюють за принципом прямого розподілення, але так як в такій системі сигнали проходять через мережу лише в одному напрямку, а циклічний зворотний зв'язок неможливий, даний вид нейронної мережі не підходить для вирішення завдання прогнозу. Тому для цієї задачі обрано RNN мережу.

В рекурентній нейронній мережі рух сигналів відбувається в обох напрямках, створюючи цикли. Завдяки цьому, мережа є потужною та динамічною, і може приймати складну структуру. Один із різновидів рекурентних мереж є штучна мережа довгої короткострокової пам’яті, завдяки своїй структурі, майже бездоганно оперує багатоваріантними даними, що ідеально підходить для передбачення такого мінливого параметру, як температура.

* 1. **Пошкодження у нейронних мережах**

Глибокі нейронні мережі складаються з безлічі окремих нейронів, які поєднують складні та інтуїтивні способи вирішення широкого кола складних завдань.Ця складність надає нейронним мережам свою силу, але із-за своїх властивостей, нейронні мережі вважають чорними ящиками.

Для побудови нових більш складних архітектур, проводяться різноманітні дослідження у ролі окремих нейронів. Задля покращення надійності системи, проводиться дослідження впливу шкоди на нейронну мережу, для визначення важливості окремих нейронів та зв’язків між ними. На рис.1.4. наведено структуру нейронної мережі з відключеним нейроном, пусті кружечки символізують нейрон без специфічної функції активації.



Рис.1.4. Відключення нейрону

Виконується штучне пошкодження слоїв нейронної мережі, а саме:

1. Пошарове пошкодження. Редукція нейронів в одному шарі.
2. Комбінаційне пошкодження. Редукція нейронів в декількох шарів одночасно. Рис.1.5 демонструє структуру нейронної мережі для пошкодження нейронів.



Рис.1.5 Структура моделі з новоствореними шарами Lambda.

При програмному розроблені, прописуються 2 шари, функції активації яких, є Lambda. Функція активації Lambda:

** (1.8)

де — вихідний сигнал попереднього шару;

Вихідний сигнал попереднього шару потрапляє на шар Lambda та обнуляється. В процесі програмної розробки, можливо виділити певний процент нейронів або специфічний нейрон для пошкодження. В розробленій існує можливість виключення нейрона с певного місця.

Для оцінки важливості нейрона, вимірюється зміна точності мережі та аналізується її продуктивність, при видаленні нейрона. Якщо нейрон є дуже важливим, його видалення має бути дуже пошкоджуючим і суттєво знизити продуктивність мережі, тоді як видалення незначного нейрона має незначний вплив.

**РОЗДІЛ 2. РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ**

* 1. **Опис роботи нейронної мережі**

Нейронна мережа тренується на даних, які зібрані з метеостанції авіабази Києврес, Бельгія. Завантажені із загальнодоступної бази наборів даних із Надійного прогнозу, rp5.ru. Отримані дані, зняті інтервалом в 10 хвилин 4,5 місяці починаючи з 01.01.2019 року. В базі даних присутні такі параметри: температура наколишнього середовища [°C], атмосферний тиск [мм.рт.ст.], вологість у [%], освітлювальність [лм], швидкість вітру [м/с], видимість [%]. Дані для тренування наведені в Додатку А.

В якості мови був обраний Python. Дана мова інтерпретується. Інтерпретатор мови представлений на всіх основних платформах: MacOS, Windows, Linux. Python є високорівневою, що дозволяє вести швидку розробку, але при цьому не погіршує читаність коду. Мова має чіткий і послідовний синтаксис, продуману модульність і масштабованість, завдяки чому, вихідний код, що написаний на Python легко читаємий.

Нейронну мережу розроблено в середовищі GoogleColaboratory.Colaboratory - це хмарна платформа від Google для просування технологій машинного навчання, що надає безкоштовну віртуальну машину з встановленими популярними бібліотеками. В даному середовищі можна створювати, навчати, використовувати мережу, а також варіювати її параметри.

Алгоритм роботи мережі:

1) Завантаження бібліотек, які потрібні для функціювання нейронної мережі: Pandas, Keras, Scikit-learn, matplotlib.

2) Обробка даних. Так як в базі даних заданий великий обсяг даних, в датасеті присутні пусті значення.Для очищення пустих комірок таблиці використовується метод заміни відсутніх значень, шляхом прямого заповнення їх попередніми моментами часу (наприклад, заповнення відсутнього значення температури останнім зафіксованим значенням температури). В програмній реалізації, ця процедура виконана за допомогою команди dropnan.

3) Розподіл даних для нейронної мережіна три набори: для навчання 60%, перевірки 20% та тестування 20%. Дані для перевірки потрібні для оцінювання якості навчання. Для перевірки вже навченої нейронної мережі застосовують дані для тестування [12].

1. Нормування значень. Так як усі дані з бази для навчання лежать в різному діапазоні значень, проводиться нормалізація даних, тобто, всі значення масштабуються від 0 до 1, окрім вибірки для тестування. В програмній розробці це виконується викликом команди MinMaxScaler.
2. Перевірка якості прогнозу, з використання середньоквадратичної функції втрат, в програмі MeanSquaredError, яка розраховується за формулою:

 (3.1)

На рис.3.1 зображені дві криві: крива блакитного кольору показує значення прогнозу, а помаранчева крива — заданого значення, в певний проміжок часу. Для кращої демонстрації, було взяті дані на інтервалі однієї години.



Рис. 2.1 Графік кривих прогнозу та заданих значень

Середньоквадратична помилка для розробленої нейронної мережі складає 0.08. Тобто, точність прогнозу складає приблизно 98,74% , що стверджує про коректну роботу нейронної мережі.

**РОЗДІЛ 3. ДОСЛІДЖЕННЯ НЕЙРОННОЇ МЕРЕЖІ З ПОШКОДЖЕННИМИ ЗВ’ЯЗКАМИ**

* 1. **Пошкодження першого шару**

Пошкодження нейронної мережі проводиться з метою оцінювання надійності системи. Крім оцінювання похибки прогнозу, проводиться аналіз тенденції наслідування зміни кривої прогнозу нормально функціонуючої системи від пошкодженої. З теоретичних роздумів, чим більше ураження нейронної системи — тим недостовірніший прогноз моделі. Кількість уражених нейронів вибирається на власний розсуд.

З метою імітації пошкодження нейронної мережі, прописуються два додаткові шари з функцією активації Lambda. Перший шар в розробленій нейронній мережі складається з 64 нейронів.В процесі виконання роботи, при пошкодженні першого шару, була здійснена поступова редукція кількості нейронів: 2, 16, 25, 32, 45, 56, 63.Запуски пошкоджених нейронних мереж проводилися при різних значеннях кількості нейронів, інші ж параметри нейронної мережі не були змінені.Розрахована середньоквадратична похибка для усіх пошкоджень, наведена в таблиці 3.1.

З рис. 3.1, не зважаючи на пошкодження 2х нейронів, отримані значення прогнозу мають відносно невелику MSE та слідують тенденції зміни кривої непошкодженої мережі. На рис.3.2., при ураженні 16-ти нейронів, мережа добре прогнозує, хоча похибка більша ніж при пошкодженні 2х нейронів. При ураженні 32 нейронів, мережа добре прогнозує локальні мінімуми та максимуми, а проміжні значення згладжує. Це свідчить про те, що при пошкодженні 50% відсотків шару, система буде якісно прогнозувати граничні значення.

З рис.3.3, видно, що система реагує лише на локальні мінімуми, що для деяких задач прогнозування, може бути прийнятним. При пошкодженні 63 нейронів, як на рис.3.4, мережа не працює зовсім.



Рис.3.1. Графік кривих після редукції 2х нейронів в першому шарі



Рис.3.2. Графік кривих після пошкодження 16 нейронів в першому шарі



Рис.3.3. Графік кривих після пошкодження 45 нейронів в першому шарі



Рис.3.4. Графік кривих після пошкодження 63 нейронів в першому шарі

В табл. 3.1 наведено середньоквадратичну похибку для пошкоджень в шарі.

Таблиця 3.1

Середньоквадратична похибка для пошкоджень в шарі

|  |  |  |
| --- | --- | --- |
| **Кількість уражених нейронів** | **MSE** | **% пошкодженої мережі** |
| 2 | 0,02 | $≈$3% |
| 16 | 2,1 | 25% |
| 32 | 4,7 | 50% |
| 45 | 8,0 | $$ 70\%$$ |
| 56 | 14,0 | 87,5% |
| 63 | 32 | $$≈98\%$$ |

* 1. **Пошкодження другого шару мережі**

Пошкодження нейронів в другому шарі мережі відбувалось аналогічним чином, але так як другий шар мережі вдвічі менший, кількість пошкоджених нейронів була змінена на: 2, 8, 16, 31. В табл. 3.2. наведено вплив пошкодження на MSE.

Таблиця 3.2.

Таблиця впливу пошкодження на MSE.

|  |  |  |
| --- | --- | --- |
| **Кількість уражених нейронів** | **MSE** | **%****пошкодженої мережі** |
| 2 | 0,2 | 6,25% |
| 8 | 9,68 | 25% |
| 16 | 14,89 | 50% |
| 30 | 31,01 | 93,75% |

Аналізуючи таблицю, проводивши аналогію з пошкодженням першого шару, виявлено, що другий шар більш чутливий до пошкоджень. Це обумовлено різною структурою нейронів в шарах, загальною кількістю нейронів і їх функцією активації. Функція активації прихованого шару— ReLU, а LTSM — гіперболічний тангенс, який має діапазон значень від -1 до 1. Результати пошкодження демонструють, що при 94% пошкоджень шару (30 нейронів), система перестає працювати зовсім, так як не реагує на будь-які зміни кривої діючих значень. Це підтверджує таблиця 3.2, при поступовому пошкодженні мережі, середньоквадратична похибка зростає.

* 1. **Комбінаційне пошкодження**

Під комбінаційним пошкодженням мається на увазі одночасне пошкодження першого і другого шару. Для дослідження впливу комбінаційного пошкодження експеримент розділяється на 3 частини: 1) однаковий відсоток пошкодження шарів; 2) ураження першого шару втричі більша ніж другого; 3) ураження другого шару втричі більша ніж першого.

Для експерименту 1, вибірка пошкоджених нейронів становить: 10%, 25%, 50%, 75%. Нижче наведені графічні зображення впливу пошкоджень на прогноз на рис. 3.5, 3.6, 3.7, 3.8.



Рис.3.5. Графік кривих після редукції 10% нейронів в обох шарах



Рис.3.6. Графік кривих після пошкодження 25% нейронів в обох шарах



Рис.3.7.Графік кривих після пошкодження50% нейронів в обох шарах



Рис.3.8. Графік кривих після пошкодження 75% нейронів в обох шарах

В процесі проведення експерименту 1, виявлено, що комбінаційне пошкодження впливає на систему ефективніше ніж пошарове.

Рис. 3.5 зображує пошкодження 10% першого і другого шару одночасно, з рисунку видно, що крива прогнозу майже співпадає з кривою діючих значень, тобто пошкодження майже не впливає на мережу. При 25% відсотках ураження, система починає гірше прогнозувати локальні мінімуми, як видно з рис.3.6. З рис.3.7, при пошкодженні 50% в обох шарах система перестає працювати коректно, крива прогнозу змінюється, але не суттєво.

Рис.3.8 демонструє, що при 75% пошкоджень в обох шарах, система перестає працювати зовсім.

Для експерименту 2, наведені графічні зображення на рис. 3.9, 3.10, 3.11.



Рис.3.9. Графік кривих після пошкодження 30% в першому шарі і 10% в другому



Рис.3.10. Графік кривих після пошкодження60% в першому і 20% в другому



Рис.3.11. Графік кривих після пошкодження 90% в першому шарі в 30% в другому

Результати другого експерименту, демонструють, що при такій комбінації пошкоджень, система працює. Рис. 3.11 демонструє, що навіть при 90% пошкоджень в першому шарі і 30% в другому, система реагує на вхідні сигнали, але якість прогнозу неприйнятна.

При проведені експерименту 3, отримано графічні зображення, які наведені нижче на рис. 3.12, 3.13, 3.14.



Рис.3.12. Графік кривих після пошкодження 10% в першому шарі в 30% в другому



Рис.3.13. Графік кривих після пошкодження 20% в першому шарі в 60% в другому



Рис.3.14. Графік кривих після пошкодження 30% в першому шарі в 90% в другому

З рис.3.13 та рис.3.14 пошкодження мали не суттєвий вплив. Порівнюючи рис. 3.14 та рис.3.11, видно, що у випадку комбінації 30%-90% система чітко демонструє локальні максимуми та мінімуми, працездатність такої системи вища.

# **ВИСНОВКИ**

 В роботі реалізована нейронна мережа для прогнозування температури навколишнього середовища, вибір структури мережі обґрунтований, теоретичний опис розробленої нейронної мережі представлений. Реалізована мережа має гарну точність, близько 98%.

Проведено пошкодження мережі : 1-го шару, 2-го шару та комбінацій обох шарів. Досліджено вплив пошкоджень на працездатність мережі. Продемонстровано граничні умови роботи мережі. Для оцінювання якості прогнозу використана середньоквадратична помилка. Наведено графічні зображення впливу пошкодження на працездатність мережі, наведені таблиці середньоквадратичних помилок.

**ПЕРЕЛІК ВИКОРИСТАНОЇ НАУКОВО-ТЕХНІЧНОЇ ЛІТЕРАТУРИ**

1. Lin Feng. Time Series Forecasting with Neural Networks / Lin Feng, Yu Xing Huo, Gregor Shirley // J. of Complexity International 2. 1995. [Електронний ресурс]. – Режим доступу: <http://journal-ci.ccse.monash.edu.au/ci/vol02/cmxhk/cmxhk.htm>
2. K. Kawakami, “Supervisedsequencelabellingwithrecurrentneural networks,” Ph.D. dissertation, Ph. D. thesis, Technical University of Munich, 2008.
3. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv preprint arXiv:1412.3555, 2014.
4. Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. PMID 9377276. doi:10.1162/neco.1997.9.8.1735
5. Hong S.G., Kim S.W. and Lee J.J., 2015. The Minimum Cost Path Finding Algorithm Using a Hopfield Type Neural Network, Proceedings IEEE International Conference on Fuzzy Systems 4, 719–726. Проблемы электротехники и электротехнологии (ПЭЭЭ-2017) Пятая Всероссийская науно-техническая конференция, Тольяттинский государственный университет, Институт энергетики и электротехники, 258-263.
6. Маряхин Е.В., Канаев Д.Г., Черненко Ю.В., 2017. Система мониторинга и прогнозирования электроэнергии, Проблемы электротехники и электротехнологии (ПЭЭЭ-2017), пятая Всероссийская научно-техническая конференция, Тольяттинский государственный университет, Институт энергетики и электротехники, 258-263.
7. QuesadaA. 5 algorithmstotrainaneuralnetwork [Електронний ресурс] / AlbertoQuesada – Режим доступу до ресурсу: <https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network>.
8. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. М.: Мир, 1992
9. Месюра В.І., Ваховська Л.М. Основи проектування систем штучного інтелекту: Навчальній посібник. — Вінниця: ВДТУ, 2000. — 96 с. — укр. мовою.
10. Ясницкий Л.Н. Введение в исскуственый интеллект, с. 70-72
11. Шолле, Ф. Глубокое обучение на Python: справ.пособие / Ф. Шолле. – СПб.: Питер, 2018. – 400 с.
12. N. Bartyzal. Artificial neural networks // Режимдоступу: http://shodhganga.inflibnet.ac.in/bitstream/10603/48/6/chaper%204\_c%20b %20bangal.
13. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления: учеб. пособие для вузов. Москва: Высшая школа, 2002. 183 с.
14. Горбань А.Н. Нейронные сети на персональном компьютере. Новосибирск: Наука, 2006. 200-230 c.
15. Лисе А.А., Степанов М.В. Нейронные сети и нейрокомпьютеры: учеб. пособие. ГЭТУ. – СПб. 2009. 64 с.
16. Standford: Natural language processing with deep learning
17. Coursera: Recurrent neural network model.
18. Fundamentals of deep learning – introduction to recurrentneural networks
19. Pushparaja Murugan and Shanmugasundaram Durairaj. Regularization andoptimization strategies in deep convolutional neural network. arXiv preprintarXiv:1712.04711, 2017.

**ANNOTATION**

The thesis presents a review of scientific and technical literature on neural networks, their properties and structure.A neural network was created to predict the ambient temperature, training was performed using data at an interval of 4.5 months, and individual neurons were removed. Afterwards, the analysis was performed on the basis of the obtained dependences, the corresponding graphs are given. The forecast error was also investigated.

According to the obtained dependences and the results of the study of ambient temperature forecast errors, the algorithm of neural network creation and training was presented in the thesis. The results of the dependence of the prediction error of the ambient temperature value on the number of neurons in the network are described. The dependences of the temperature prediction error on the number of neurons in the layer were investigated, performed for two cases: in case of layer - by - layer damage and in case of combination damage of the layer. The neural network was tested on training data.

**Keywords**: neural network; LSTM; forecasting; damage of neurons;

**ДОДАТОК А**

**ЛІСТИНГ ПРОГРАМИ**

**Програмний код розробленої нейронної мережі**

from math import sqrt

from sklearn.model\_selection import train\_test\_split

import numpy as np

from numpy import concatenate

from pandas import read\_csv

from pandas import DataFrame

from pandas import concat

from pandas import to\_datetime

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean\_squared\_error

from keras.models import Sequential

from keras.layers import LSTM

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Lambda

from keras.losses import MeanSquaredError

from matplotlib import pyplot

def series\_to\_supevised(dataset, n\_in=1, n\_out=1, dropnan=True):

    num\_vars = 1 if type(dataset) is list else dataset.shape[1]

    dataframe = DataFrame(dataset)

    cols, names = list(), list()

    # input sequence (t-n, ....t-1)

    for i in range(n\_in, 0, -1):

        cols.append(dataframe.shift(i))

        names += [('var%d(t-%d)' % (j+1, i)) for j in range(num\_vars)]

 # forecast sequence (t, t+1 .... t+n)

    for i in range(0, n\_out):

        cols.append(dataframe.shift(-i))

        if i == 0:

            names += [('var%d(t)' % (j+1)) for j in range(num\_vars)]

        else:

            names += [('var%d(t+%d)' % (j+1, i)) for j in range(num\_vars)]

     # put it all together

    agg = concat(cols, axis=1)

    agg.columns = names

    # drop rows with NaN values

    if dropnan:

        agg.dropna(inplace=True)

    return agg

def split\_window(dataset):

  dataset

dataset = read\_csv("drive/My Drive/Data/KAG\_energydata\_complete.csv")

dataset.drop(dataset.columns[0:21], axis=1, inplace=True)

dataset.drop(dataset.columns[6:8], axis=1, inplace=True)

cols = dataset.columns.tolist()

cols = cols[1:] + cols[:1]

dataset = dataset[cols]

dataset

19735 rows × 6 columns

values = dataset.values

#values[:,0] = to\_datetime(values[:,0])

scaler = MinMaxScaler(feature\_range=(0,1))

scaled = scaler.fit\_transform(values)

values.shape

Index(['Press\_mm\_hg', 'RH\_out', 'Windspeed', 'Visibility', 'Tdewpoint',

 'T\_out'],

dtype='object')

reframed = series\_to\_supevised(scaled, 5, 1)

#reframed = series\_to\_supevised(scaled, 1, 1)

# drop columns we don't want to predict

reframed.drop(reframed.columns[-6:-1], axis=1, inplace=True)

print("reframed: ", reframed.shape, "reframed head: ", reframed.head())

# split into train and test sets

values = reframed.values

X = values[:,:-1]

Y = values[:,-1]

#Y2 = dataset[:,16]

# Split Data to Train and Test

X\_Train, X\_Secondary, Y\_Train, Y\_Secondary = train\_test\_split(X, Y, test\_size=0.4)

X\_Test, X\_Val, Y\_Test, Y\_Val = train\_test\_split(X\_Secondary, Y\_Secondary, test\_size=0.5)

# reshape input to be 3D [samples, timesteps, features]

X\_Train = X\_Train.reshape((X\_Train.shape[0], 1, X\_Train.shape[1]))

X\_Test = X\_Test.reshape((X\_Test.shape[0], 1, X\_Test.shape[1]))

X\_Val = X\_Val.reshape((X\_Val.shape[0], 1, X\_Val.shape[1]))

import keras.backend as K

nums1 = np.ones(64)

nums2 = np.ones(32)

mask1 = K.variable([nums1])

mask2 = K.variable([nums2])

model = Sequential()

model.add(LSTM(64, input\_shape=(X\_Train.shape[1], X\_Train.shape[2])))

model.add(Lambda(lambda x: x \* mask1))

model.add(Dense(32, activation='relu'))

model.add(Lambda(lambda x: x \* mask2))

model.add(Dense(1))

model.compile(loss='mse', optimizer='adam')

# fit

history = model.fit(X\_Train, Y\_Train, epochs=10, batch\_size=10, validation\_data=(X\_Test, Y\_Test), verbose=2, shuffle=False)

# nums1 = np.ones(64)

# nums1[:10] = 0

# np.random.shuffle(nums1)

nums1 = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]

K.set\_value(mask1,[nums1])

# nums2 = np.ones(32)

# nums2[:10] = 0

# np.random.shuffle(nums2)

nums2 = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]

K.set\_value(mask2,[nums2])

# X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

# Using 'none' reduction type.

mse = MeanSquaredError()

mse(Y\_Val, model.predict(X\_Val)).numpy()

pyplot.plot(history.history['loss'], label='Train')

pyplot.plot(history.history['val\_loss'], label='Test')

pyplot.legend()

pyplot.show()

X\_Graph = X.reshape((X.shape[0], 1, X.shape[1]))

# Pred\_Graph = temp\_scaler.inverse\_transform(model.predict(X\_Graph))

# Y\_Graph = temp\_scaler.inverse\_transform([Y])

#pyplot.yticks(np.arange(-5.5, 26.5, 0.005))

pyplot.plot(model.predict(X\_Graph), label='Predicted')

pyplot.plot(Y, label='Actual')

pyplot.legend()

pyplot.show()

import pandas as pd

Pred\_Graph = temp\_scaler.inverse\_transform(model.predict(X\_Graph))

Y\_Graph = temp\_scaler.inverse\_transform([Y])

Y\_Graph = Y\_Graph.reshape((Y\_Graph.shape[1], 1))

res = pd.DataFrame(data=Pred\_Graph, columns=['Prediction'])

res['Actual'] = Y\_Graph

res.plot()

res.to\_csv("drive/My Drive/Data/Temperature\_result.csv")