НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Приладобудівний факультет

Кафедра виробництва приладів

До захисту допущено:

Завідувач кафедри

________ Віктор АНТОНЮК

«___»____________2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп’ютерно-інтегровані технології виробництва приладів»

спеціальності 151 «Автоматизація та комп’ютерно- інтегровані технології»

на тему: «Моделювання технологічних процесів засобами штучних нейронних мереж»

Виконала:
студентка IV курсу, групи ПБ-61
Соколова Олена Андріївна ________

Керівник:
Доцент, к.т.н. доцент
Вислоух Сергій Петрович ________

Рецензент:
асистент, к.т.н.
Котляр Світлана Сергіївна ________

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка ____________

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ДР 61.17. 1720.000 ПЗ	Пояснювальна записка	106	
3	А1	ДР 61.17. 1720.001 СХ	Методи моделювання процесів і систем	1	
4	А1	ДР 61.17. 1720.002 СХ	Методи математичного моделювання	1	
5	А1	ДР 61.17. 1720.003 СХ	Засоби математичного моделювання	1	
6	А1	ДР 61.17. 1720.004 СХ	Можливості моделювання засобами TensorFlow	1	
7	А1	ДР 61.17. 1720.005 СХ	Програмні модулі TensorFlow	1	
8	А1	ДР 61.17. 1720.006 СХ	Алгоритм моделювання засобами TensorFlow	1	
9	А1	ДР 61.17. 1720.007 СХ	Практичне моделювання засобами TensorFlow	1	
10	А1	ДР 61.17. 1720.008 СХ	Практичні рекомендації	1	
11	А1	ДР 61.17. 1720.009 СХ	Висновки	1	

	ПІБ	Підп.	Дата	ДР ПБ61.17.1720.000.00		
Розробн.	Соколова О.А.			Відомість дипломної роботи	Лист	Листів
Керівн.	Вислоух С.П.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. ВП Гр. ПБ-61	
Н/контр.						
Зав.каф.	Антонюк В.С.					

Пояснювальна записка
до дипломної роботи
на тему: «Моделювання технологічних процесів засобами штучних нейронних мереж»

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра виробництва приладів

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «**Автоматизація та комп’ютерно- інтегровані технології**»

Освітньо-професійна програма «**Комп’ютерно-інтегровані технології виробництва приладів**»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_______ Віктор АНТОНЮК
«___»_____________2020 р.

ЗАВДАННЯ

на дипломну роботу студентці

Соколовій Олені Андріївні

1. Тема роботи «**Моделювання технологічних процесів засобами штучних нейронних мереж**», керівник роботи Вислоух Сергій Петрович, к.т.н., доцент, затверджені наказом по університету від «25» травня 2020 р. № 1180-с

2. Термін подання студентом роботи 30 травня 2020 р.

3. Вихідні дані до роботи Основи побудови штучних нейронних мереж та опис засобів, що їх реалізують.

4. Зміст роботи

1. Літературний огляд за темою дипломної роботи та постановка задачі досліджень.
2. Штучні нейронні мережі для моделювання процесів і систем.
3. Засоби моделювання, що використовують ШНМ.
4. Моделювання процесів і систем засобами TensorFlow.
5. Практичне застосування TensorFlow для моделювання технологічних процесів і систем.
6. Розроблення практичних рекомендацій з використання результатів дослідження.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1 Методи моделювання процесів і систем.

2 Методи математичного моделювання технологічних процесів та їх параметрів. Штучні нейронні мережі.

3 Засоби математичного моделювання процесів і систем з використанням ШНМ.

4 Можливості моделювання засобами TensorFlow.

5 Програмні модулі TensorFlow.

6 Методика застосування TensorFlow.

7 Практичне моделювання параметрів та процесів засобами TensorFlow .

8 Практичні рекомендації з використання результатів досліджень.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 14 березня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Літературний огляд за темою дипломної роботи та постановка задачі досліджень	13 квітня 2020 р.	
2	Огляд штучних нейронних мережі для моделювання процесів і систем	20 квітня 2020 р.	
3	Аналіз засобів математичного моделювання процесів і систем з використанням ШНМ	27 квітня 2020 р.	
4.	Моделювання параметрів і систем засобами TensorFlow	4 травня 2020 р.	
5.	Практичне моделювання параметрів та процесів засобами TensorFlow	11 травня 2020 р.	
6	Розроблення практичних рекомендацій з використання TensorFlow для моделювання процесів і систем	25 травня 2020 р.	
7	Оформлення пояснювальної записки та ілюстративної частини дипломної роботи	30 травня 2020 р.	

Студент Олена СОКОЛОВА

Керівник Сергій ВИСЛОУХ

АНОТАЦІЯ

Метою дипломної роботи є моделювання технологічних процесів засобами штучних нейронних мереж.

Дипломна робота складається з: пояснювальної записки, що містить 106 сторінок, 13 таблиць, 32 рисунки та графічної частини, що включає 9 ілюстрації формату А1.

В даній дипломній роботі розв'язуються задачі моделювання технологічних процесів засобами штучних нейронних мереж, вибору інструменту моделювання, практичного застосування платформи TensorFlow для моделювання технологічних процесів для прогнозування параметрів.

В пояснювальній записці дипломної роботи представлено 4 розділи: літературний огляд за темою дипломної роботи та постановка задачі досліджень, штучні нейронні мережі для моделювання процесів та систем, моделювання процесів і систем засобами TensorFlow, практичне застосування TensorFlow для моделювання технологічних процесів і систем.

В першому розділі розглянуто загальні відомості про методи моделювання та обрано найбільш доцільного метод для моделювання процесів та систем.

В другому розділі описано сутність моделювання засобами штучних нейронних мереж та області їх використання, а саме: застосування ШНМ в технології приладобудування.

В третьому розділі розглянуто області використання платформи TensorFlow, її структуру та розроблено алгоритм моделювання засобами TensorFlow.

В четвертому розділі розглянуто питання створення штучної нейронної мережі за допомогою платформи TensorFlow та проведено моделювання технологічних процесів обробки різних матеріалів. Крім того, в роботі надано рекомендації до застосування розроблених програм.

Графічна частина містить схеми методів моделювання, засобів математичного моделювання, можливостей моделювання засобами TensorFlow, програмних модулів TensorFlow, алгоритм застосування TensorFlow.

ANNOTATION

The graduation thesis goal is the modeling of technological processes through artificial neural networks.

Thesis consists of: an explanatory note containing 106 pages, 13 tables, 32 drawings and a graphic part, which includes 9 illustrations in A1 format.

This graduation thesis solves the problems of modeling technological processes through artificial neural networks, the choice of modeling tool, the practical application of the TensorFlow platform for modeling technological processes to predict parameters.

The explanatory note of the graduation thesis presents 4 sections: literature review on the topic of the graduation thesis and research problem statement, artificial neural networks for modeling processes and systems, modeling of processes and systems by TensorFlow, practical application of TensorFlow for modeling processes and systems.

The first section reviews general information about modeling methods and selects the most appropriate method for modeling processes and systems.

The second section describes the essence of modeling through artificial neural networks and the area of their usage, namely: the use of ANN in instrument-making technology.

The third section reviews the areas of use of the TensorFlow platform and its structure. The algorithm of modeling with TensorFlow usage has been developed.

The fourth section considers the creation of an artificial neural network using the TensorFlow platform and simulates the technological processes of various materials processing. In addition, the recommendations for the developed programs usage have been provided in the graduation thesis.

The graphic part contains the schemes of modeling methods, of mathematical modeling means, of modeling possibilities by TensorFlow means, of TensorFlow software modules and the TensorFlow application algorithm.

ЗМІСТ

ТЕРМІНИ ТА СКОРОЧЕННЯ

Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) – повністю автоматизований публічний тест Тюрінга.

CPU (Central Processing Unit) – центральний процесор.

GPU (Graphics Processing Unit) – графічний процесор.

ТПВ – технологічна підготовка виробництва.

ШНМ – штучна нейронна мережа.

ТП – технологічний процес.

IoT (Internet of Things) – інтернет речей.

ТЗ – технічне завдання.

МГВА – метод групового врахування аргументів.

PSP (Post-Synaptic Potential) – пост-синаптичний потенціал.

SGD (Stochastic Gradient Descent) – стохастично-градієнтний спуск.

UX / UI – дизайн взаємодії з користувачем.

ZTS (Zero Trust Security) – архітектура, що відноситься до концепцій безпеки, які не передбачають автоматичну довіру до користувача.

API (Application Programming Interface) – програмний інтерфейс додатку, інтерфейс прикладного програмування.

CUDA – архітектура паралельних обчислень.

CIFAR-10 – набір зображень для навчання алгоритмів машинного навчання та комп’ютерного зору.

CRM – концепція взаємодії, що передбачає об’єднання розрізнених інструментів ведення бізнесу в налагоджену систему.

Мережа ART – мережі на базі теорії адаптивного резонансу.

Мережа SOF – самоорганізуючі мапи, мережі Кохонена.

Мережа RBF – мережі з радіальними базисними функціями.

ВСТУП

Приладобудування – це галузь машинобудування, що займається розробкою та випуском пристроїв регулювання, засобів вимірювання, аналізу та обробки інформації, зокрема засобів автоматизації та систем керування. Наразі є потреба у нових, більш сучасних технологіях, оскільки постійно зростаючий попит на продукцію та, як наслідок, потреба у підвищенні ефективності приладобудівного виробництва гостро ставить задачу скорочення строків технологічної підготовки виробництва та покращення результатів.

В приладобудуванні важливим та базовим процесом є обробка матеріалів. Деякі матеріали важко піддаються обробці, що може призвести до різних наслідків: руйнування ріжучої кромки інструменту, деформація деталі, зниження шорсткості деталі – усе це призводить до браку. Тому в сучасному приладобудуванні важливою є задача вибору раціональних параметрів обробки певного матеріалу ще на етапі технологічної підготовки виробництва. Дану задачу можна вирішити за допомогою методів моделювання, прогнозування, знаходження закономірностей, машинної обробки даних та аналізу результатів тощо.

Значного прогресу завдяки механізації та автоматизації досягла і галузь обробки матеріалів. Наразі штучні нейронні мережі застосовуються для контролю якості продукції, оптимізації режимів виробничого процесу, попередження аварійних ситуацій на виробництві тощо.

Метою дипломної роботи є отримання практичних навичок у виборі найбільш доцільних методів моделювання для вирішення конкретної задачі та у створенні програм, що моделюють технологічний процес, для прогнозування найбільш прийнятних та раціональних параметрів обробки.

РОЗДІЛ 1.
ЛІТЕРАТУРНИЙ ОГЛЯД ЗА ТЕМОЮ ДИПЛОМНОЇ РОБОТИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕНЬ

1.1 Методи моделювання процесів і систем

Підвищення ефективності в приладобудуванні потребує якісного вирішення задач технологічної підготовки виробництва на основі визначення раціональних або оптимальних параметрів технологічних процесів. Тому важливою є задача прогнозування вихідних технологічних параметрів за поточними даними про виробничий процес. На сьогодні це здійснюється шляхом моделювання цих параметрів, адже метою моделювання є отримання, обробка, представлення і використання інформації про об'єкти, які взаємодіють між собою і зовнішнім середовищем [1].

Методом моделювання описуються структура об'єкта (статична модель), процес його функціонування і розвитку (динамічна модель). У моделі відтворюються властивості, зв'язки, тенденції досліджуваних систем і процесів, що дає змогу оцінити їх стан, зробити прогноз, прийняти обґрунтоване рішення [2]. Можливості моделювання, тобто перенос результатів, отриманих в ході дослідження моделі, на оригінал, базуються на тому, що модель в певному сенсі відображає (відтворює, моделює, описує, імітує) деякі риси об'єкту, що цікавлять дослідника.

Моделювання можна застосовувати в різних областях науки та техніки, наприклад, для моделювання виробничих процесів та систем. Моделювання використовується на різних етапах досліджень з відповідною метою. Наприклад, на початкових етапах моделювання дозволяє отримати інформацію, що є необхідною для формулювання та уточнення роботи гіпотези.

Моделювання виробничих процесів полягає в їх дослідженні шляхом створення моделей, що відображають структуру процесів, характеристики об'єктів та потоки інформації. Моделювання виробничих процесів за допомогою математичних моделей на ЕОМ дозволяє без застосування вартісного натурного експерименту, як це часто вимагає фізичне моделювання, оцінювати характеристики спроектованих виробництв, процесів, вирішувати задачі, які виникають на стадії розробки [3].

Існує велика кількість різноманітних методів моделювання процесів та систем. На рисунку 1.1 представлено систематизоване представлення сучасних моделей, де в якості підстав для поділу прийняті призначення методу (верхня частина діаграми) та природа та особливості формалізованого представлення конкретних моделей (нижня частина діаграми) [4].

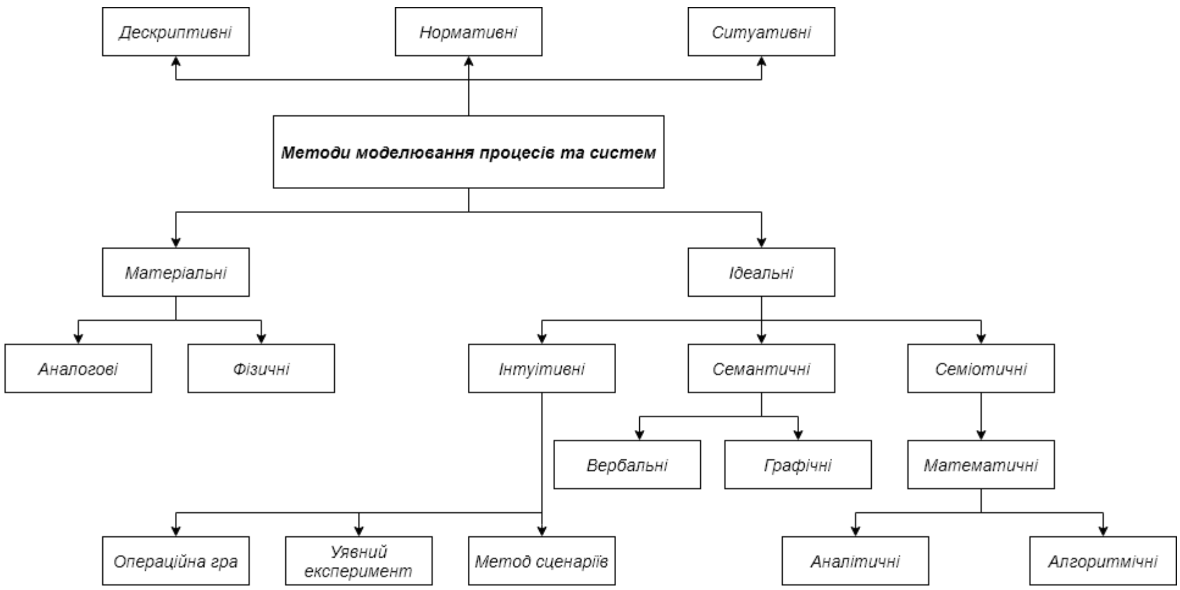


Рис. 1.1. Методи моделювання процесів та систем.

За призначенням моделювання процесів та систем поділяється на три типи:

– дескриптивне, яке пов’язане з описом властивостей модельованих систем та поясненням законів зміни їх параметрів за допомогою слів, рисунків або символів;

– нормативне, що пов’язане з пошуком відповідей на питання, як має бути, тобто як має поводити себе система. Ці моделі повинні обґрунтовувати раціональні структури та параметри модельованих об’єктів, а також визначати оптимальні траєкторії досягнення поставленої мети з урахуванням тенденцій та протиріч, що виявлені за допомогою дескриптивних методів;

– ситуативне, яке необхідне для виявлення розходження між наявними та необхідними параметрами об’єкту та для усунення факторів, що стали причиною їх виникнення. Головною перевагою даних моделей є їх придатність для виконання прогнозу відповідних ризиків заздалегідь, а також для апріорної кількісної оцінки та оптимізації заходів щодо їх зниження.

За своєю природою всі моделі можуть бути поділені на два загальних класи.

1. Матеріальні, якими зазвичай є макети, лабораторні установки, манекени, що в свою чергу поділяються на:

– фізичні моделі, які зазвичай є геометричною подобою конкретного явища. Перехід від моделі до оригіналу та навпаки вимагає перерахунку характеристик моделі та оригіналу, що еквівалентно переходу від однієї системи координат (одиниць вимірювання) до іншої. Частіше застосовуються в авіа-, автомобіле-, ракето- та суднобудуванні;

– аналогові, які є геометричною подібністю процесу. В основу даного методу покладено співпадіння математичного опису таких явищ, як механічні та електричні коливання, обтікання твердих тіл потоками газу та рідини тощо. Головною перевагою застосування аналогових моделей полягає у тому, що вивчення модельованих за їх допомогою процесів можна проводити у більш сприятливих для дослідження умовах.

2. Ідеальні моделі – це опис або представлення досліджуваних явищ за допомогою уявних, графічних та математичних образів. Методи ідеального моделювання органічно доповнюють один одного, хоча і мають іноді досить конкретні області рекомендованого використання. Інтуїтивні моделі частіше за все слугують ніби генератором ідей для наступної побудови семантичних, які є основою для семіотичних.

Під інтуїтивним моделюванням розуміють таке необґрунтоване з позиції формальної логіки представлення об’єкту дослідження, яке не підлягає формалізації або не потребує її. Виконується у вигляді уявних експериментів, сценаріїв та ігрових ситуацій – з метою оцінки та задля переналаштування до наступних дій заздалегідь.

Семантичне моделювання логічно обґрунтоване гіпотезами, які створені на базі спостереження за об’єктом-оригіналом та його аналогами. В дану групу входять:

– вербальне моделювання, що представляє сукупність висловів на уточненій природній мові;

– графічне моделювання використовує різноманітні носії інформації, на яких розміщуються причинно-наслідкові діаграми, структурно-функціональні схеми та інші малюнки.

Семіотичне (знакове) моделювання. Найбільш відомим представником даного методу моделювання є математичне моделювання, моделі якого відносяться до найбільш формалізованих. Основними класами моделей, використовуваних в математичному моделюванні, є:

– аналітичні. Головною перевагою аналітичного моделювання є можливість отримання конкретних математичних висловів, що використовують арифметичні операції та переходи від лімітів до натуральних чисел. Частковими випадками аналітичних моделей є всі коректні алгебраїчні вирази, а також та їх частина, котра має граничну кількість параметрів та використовується для отримання наближених результатів;

– алгоритмічні. Можуть враховувати практично будь-яку кількість вагомих факторів, а тому – і застосовуються для моделювання найбільш складних об’єктів на сучасних потужних комп’ютерах. Однак, у більшості випадків алгоритмічні моделі дозволяють отримувати лише наближені результати, використовуючи метод чисельного або імітаційного моделювання.

Основним методом вивчення, оптимізації та дослідження складних процесів і об’єктів, до яких відносяться технологічні процеси, є математичне моделювання. Даний метод полягає в побудові математичної моделі та дослідженні її аналітичними або числовими методами для отримання необхідної характеристики досліджуваного реального процесу або системи [5].

1.2. Методи математичного моделювання

Варто зазначити, що будь-яка модель – це умовний образ реально існуючих закономірностей, певне наближення до об’єктивної дійсності. Тому спрощення під час побудови математичних моделей, зокрема для дослідження технологічних процесів, є не тільки вимушеними, а й навмисними, оскільки одночасне охоплення всіх аспектів реальності не завжди доцільне і нерідко перевищує можливості дослідників.

Математичною моделлю об’єкту, процесу або явища називають запис його властивостей формальною мовою для отримання певних властивостей досліджуваного об’єкта шляхом використання формальних методів. Основною метою математичного моделювання процесів та систем є виділення законів в природі та техніці та їх опис за допомогою математичних формул [6].

На сьогодні математичне моделювання використовується [7]:

– з метою проведення експерименту або чисельної оцінки, тобто для передбачення наслідків зміни умов, методів використання або образу дій у випадку, коли впровадження цих змін в реальних умовах пов’язане з великими втратами;

– як засіб дослідження систем з метою їх переробки або вдосконалення;

– як засіб ознайомлення персоналу з системами або умовами, які, можливо, поки що не існують, в реальній дійсності;

– задля перевірки або демонстрації нової ідеї, системи або методу;

– як засіб передбачення майбутнього та забезпечення, таким чином, кількісної основи для планування та прогнозування.

Процес створення математичних моделей досить трудомісткий, довготривалий та пов'язаний із використанням висококваліфікованої праці. Особливістю математичних моделей, створюваних в даний час, є їх комплексність, що є наслідком складності модельованих об’єктів. В сучасному світі характерним є представлення об’єкту моделювання у вигляді системи взаємодіючих елементів, що призводить до ускладнення моделі та необхідності використання декількох теорій, застосування сучасних обчислювальних методів та обчислювальної техніки. У разі моделювання складних об’єктів неможливо задовольнити всім вимогам, що висуваються. Тож, доводиться створювати цілий спектр моделей одного і того ж реального об’єкту. Для зменшення витрат на розробку моделей та ймовірності виникнення помилок розроблено алгоритм створення математичних моделей [8].

На рисунку 1.2 представлено етапи математичного моделювання [8].

Рис. 1.2. Етапи математичного моделювання.

1. Обстеження об’єкту моделювання, формулювання завдання на розробку моделі. Основним завданням обстеження об’єкту є підготовка змістовної постановки задачі моделювання. Змістовна постановка задачі – це перелік сформульованих у словесній формі основних питань, на які повинна відповідати розроблена модель, опис її основних характеристик. Під час виконання даного етапу виконуються наступні завдання:

– виявлення основних факторів, механізмів, що впливають на роботу поведінку об’єкта, методів визначення параметрів, що описують об’єкт;

– збір та перевірка наявної експериментальної інформації про об’єкти-аналоги, проведення додаткових експериментів за необхідності;

– аналітичний огляд наявної літературної та іншої інформації, аналіз розроблених раніше моделей даного об’єкту або подібного;

– аналіз та узагальнення усього накопиченого матеріалу, розробка загального плану створення моделі;

– змістовна постановка задачі (ТЗ) на розробку моделі.

2. Розробка математичного опису об’єкту. Виконання концептуальної постановки задачі – сформованого у термінах певної науки переліку питань, що повинна вирішувати модель, а також сукупність гіпотез відносно властивостей і поведінки об’єкта моделювань. Згідно з прийнятими гіпотезами визначається набір параметрів, що описують стан об’єкта, та перелік законів, що описують поведінку об’єкта, взаємозв’язок між параметрами та об’єктом із навколишнім середовищем.

3. Вибір критерію якості об’єкту. Важливою складовою концептуальної моделі є критерій якості об’єкту. Призначення критерію – встановлення переважного варіанта виконання об’єкта в задачах оптимізації. Для масових оптимізаційних задач в системах, що знаходяться на нижчих ієрархічних рівнях, в якості критерію оптимальності обирають мінімум зведених витрат. Цей показник враховує первинні та експлуатаційні витрати в грошових показниках. Поряд з критерієм розглядаються і додаткові – екологічні обмеження, перспективність прийнятих рішень з врахуванням технічного прогресу, підвищення продуктивності праці тощо. Чим вищий рівень, тим більший вплив мають саме додаткові критерії.

4. Математична постановка задачі. Математичний опис об’єкту – це сукупність математичних співвідношень. що описують поведінку об’єкту моделювання. Найбільш простий опис – за допомогою алгебраїчних рівнянь. Проте область застосування таких моделей досить обмежена. Для створення моделей складних систем використовують знання, що сконцентровані у аксіомах, теоремах, законах, що мають чітке математичне формулювання.

Відокремлюють закони, справедливі для всіх типів задач (наприклад, закони збереження маси, кількості руху і енергії тощо) та співвідношення, що описують поведінку окремих об'єктів (наприклад, закон Гука, Стефана-Больцмана тощо), проте вони мало вивчені. Іноді необхідно розробляти власні замикаючі співвідношення, наприклад, за інтенсивністю тепловіддачі в певних умовах тощо.

В більшості випадків математичний опис об'єкту включає диференційні рівняння, диференційні рівняння у часткових похідних, інтегральні рівняння тощо. Обов'язковим елементом математичної постановки задачі є прийняті допущення та спрощення під час побудови моделі.

5. Якісний аналіз моделі, перевірка коректності моделі. Для контролю правильності складного математичного опису необхідно виконати ряд перевірок:

– контроль закономірностей;

– контроль порядків;

– контроль характеру залежностей і фізичного змісту;

– контроль екстремальних ситуацій;

– контроль граничних умов;

– контроль математичної замкнутості.

Якщо математична модель пройшла вище наведені контрольні перевірки, то вона називається коректною.

6. Обґрунтування і вибір методу розв'язання задачі. В великій мірі вибір методу розв'язання задачі залежить від кваліфікації спеціалістів.

Аналітичні методи більш зручні для аналізу результатів, але можуть бути застосовані для відносно простих моделей. Використання будь-якого аналітичного методу призводить до виникнення похибки (похибки вхідних даних, похибки вибору методу, похибки округлення тощо).

Алгоритмічні методи більш трудомісткі, вимагають знання обчислювальної математики, програмних комплексів, потужної техніки. Точність таких методів залежить від вибору метода і параметрів, наприклад, кроку інтегрування.

Підбір методу повинен забезпечувати ефективність (мінімум часу за достатньої точності), стійкість та точність результатів.

7. Реалізація моделі у вигляді програми. Процес розробки програмного продукту на основі розробленої моделі є не менш складним і важливим, ніж попередні етапи. Процес створення програмного забезпечення складається з наступних етапів:

– складання ТЗ на розробку пакета програм;

– проектування структури програмного комплексу;

– кодування алгоритму;

– тестування та відлагодження;

– супровід та експлуатація.

Найбільш ефективним варіантом є розбиття програми на окремі модулі. Для кожного модуля складається алгоритм, що дозволяє виконувати відповідні функції. Розробляється система зв'язків між модулями, яка називається схемою потоків даних програмного комплексу.

8. Перевірка адекватності моделі. Під адекватністю моделі розуміють ступінь відповідності результатів, отриманих з використанням розробленої моделі до даних експерименту або тестової задачі. Перевірка адекватності має на меті:

– впевнитись у справедливості прийнятої сукупності гіпотез на етапі концептуальної і математичної моделі;

– встановити, що точність отриманих результатів відповідає точності, вказаній у технічному завданні.

В моделях для виконання оціночних розрахунків задовільною вважається похибка 10…15%. В моделях, що використовуються в керуючих і контролюючих системах допустима похибка 1…2% і менше.

Неадекватність результатів можлива з трьох причин:

– значення заданих параметрів моделі не відповідають допустимій області цих параметрів;

– прийнята система гіпотез вірна, але константи і параметри у визначальних співвідношеннях не встановлені достатньо точно;

– невірна використана система гіпотез.

За неадекватності результатів необхідно провести коригування моделі, розглядаючи причини у вищенаведеній послідовності.

9. Практичне використання розробленої моделі.

На рис.1.3 наведено методи математичного моделювання.

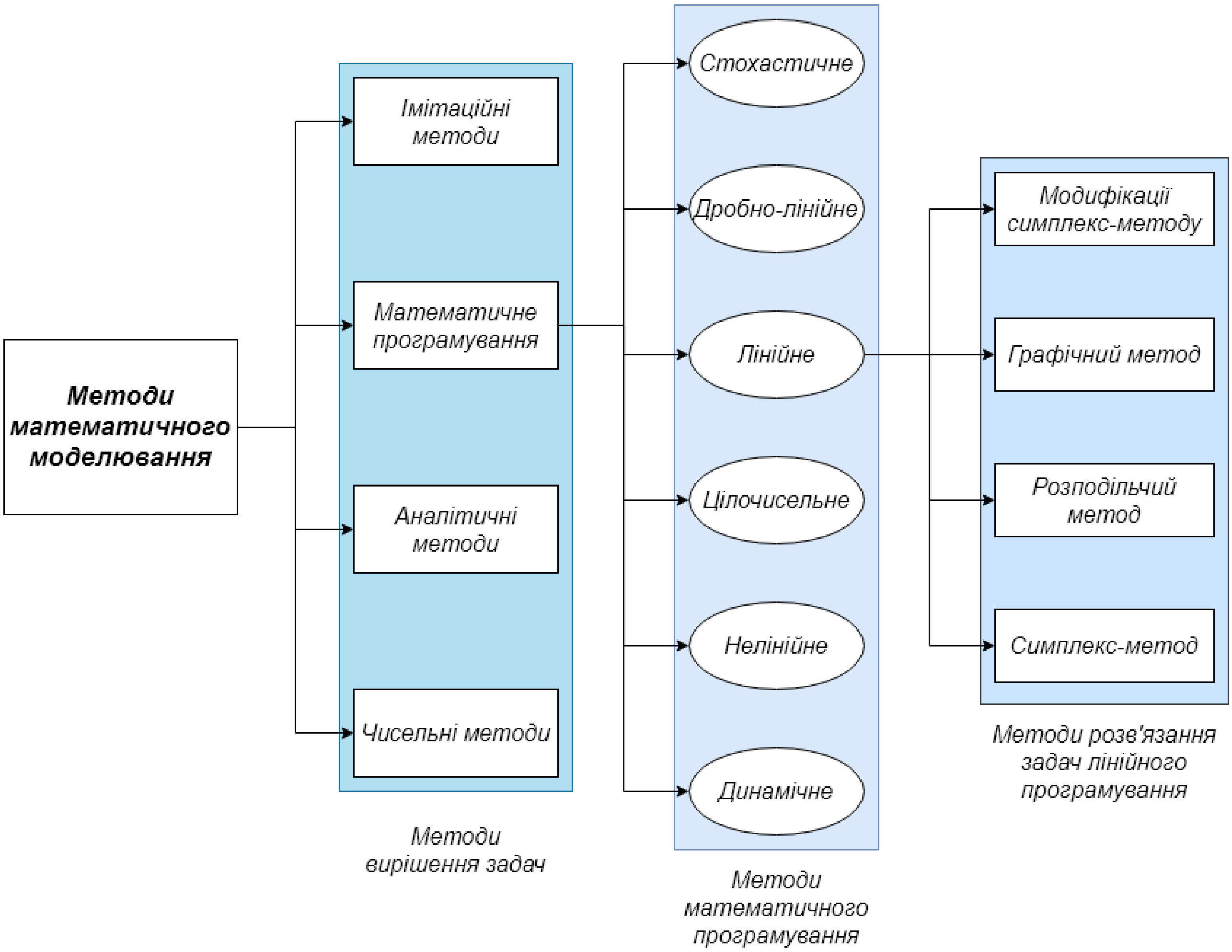


Рис. 1.3. Методи математичного моделювання.

До методів математичного моделювання відносять:

– аналітичне моделювання – метод, що використовується для моделювання, роз’яснення та прогнозування механізмів складних фізичних процесів. Для нього характерно, що процеси функціонування системи записують використовуючи функціональні співвідношення (алгебраїчні, диференційні, інтегральні рівняння) [9];

– чисельне моделювання – це процес вирішення математичної моделі чисельним методом із застосуванням ЕОМ. Даний метод пов'язаний з багатьма похибками, що обумовлені: неточністю представлення зав’язків між параметрами в математичній моделі; похибками вихідних параметрів; неточністю чисельного методу тощо [10];

– імітаційне моделювання, як інструмент експериментального дослідження складних систем, охоплює методологію створення моделей систем, методи алгоритмізації та засоби програмних реалізацій імітаторів, планування, організацію і виконання на ЕОМ експериментів з імітаційними моделями, машинну обробку даних та аналіз результатів [11];

– математичне програмування – це метод, що використовує теорію і чисельні методи вирішення багатовимірних граничних задач з обмеженнями. Це задачі на пошуки екстремумів функцій багатьох змінних з обмеженнями на область варіювання цих змінних [12].

Існує багато різноманітних класифікацій математичних моделей за різними ознаками. На рисунку 1.4 приведено класифікацію математичних моделей за трьома ознаками:

– характер зміни змінних;

– способи врахування випадковостей;

– математичний апарат.

Отже, основним методом для моделювання технологічних процесів та систем є математичне моделювання. Математична модель повинна відображати всі основні фактори, що впливають на систему або процес, а також взаємозв’язки, що якомога точніше описують реальні ситуації, критерії та обмеження. Також необхідно, аби математична модель була достатньо універсальною для опису близьких за об’єктів, та простою, аби дозволити виконувати необхідні дослідження з оптимальними витратами, проте в той же час все ще добре описувати досліджуваний процес.

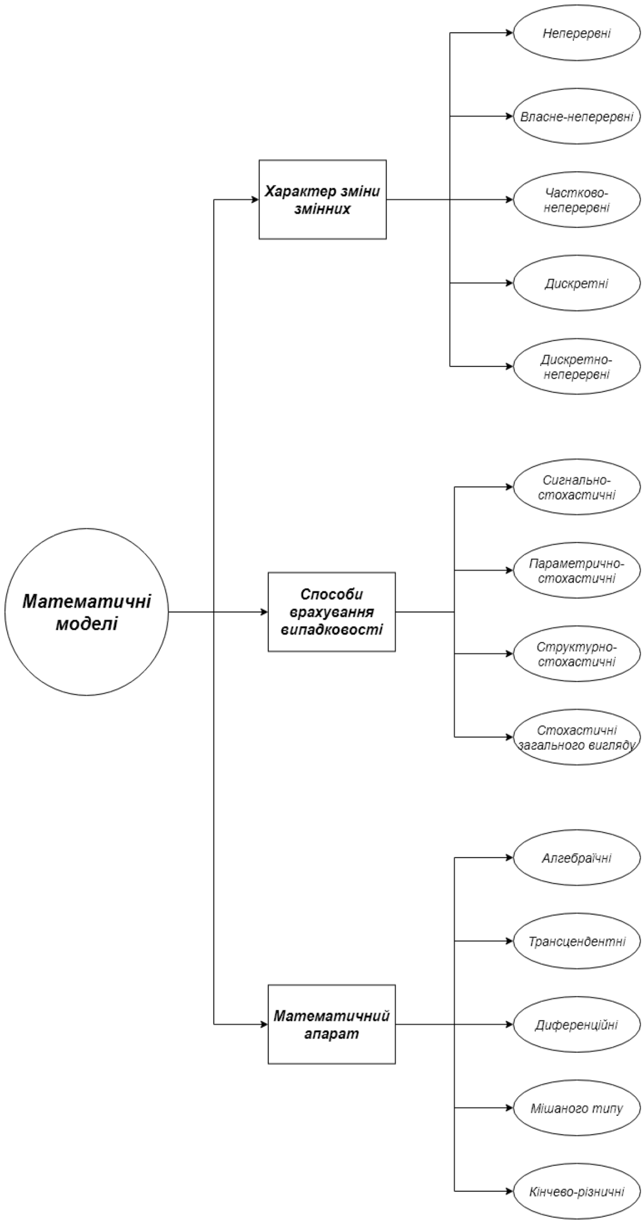


Рис. 1.4. Класифікація математичних моделей.

1.3. Математичне моделювання параметрів і процесів в приладобудуванні

Для отримання математичних моделей вихідних технологічних параметрів обробки та їх прогнозування широко використовуються регресійний аналіз, методи кореляційного, дисперсійного аналізу тощо. Проте дані методи вимагають великої кількості експериментальних досліджень для отримання результатів з необхідною точністю. Наразі є потреба у методах, що дозволять отримати модель із бажаною точністю при малій кількості експериментальних досліджень. До таких методів можна віднести евристичні методи, що дозволяють при невеликій кількості експериментів отримати якісне прогнозування вихідних технологічних параметрів. Наприклад, метод евристичної самоорганізації моделей, що запропонований академіком А.Г. Івахненком є одним з методів обробки результатів експериментальних досліджень. Для розв’язання математичних задач метод самоорганізації реалізовано як метод групового врахування аргументів (МГВА) [5].

У ряді випадків багато факторів, які необхідно враховувати при побудові моделі, мають якісний характер та не можуть бути виміряні за допомогою метричних шкал. Також варто відмітити, що в умовах слабкої математичної формалізації деяких процесів, що проходять у складних системах, та обмеженого об’єму статистичних даних зростає роль експертної інформації, що використовується в процесі моделювання. Дані обставини знижують ефективність застосування відомих варіантів реалізації МГВА для побудови математичних моделей складних систем різних типів. Також є суттєвий недолік, пов'язаний із суб’єктивністю вибору виду та параметрів функцій приналежності. Тому наразі інтерес представляє застосування інших сучасних методів моделювання.

Метод групового врахування аргументів використовується для розв’язання наступних видів задач [5]:

– ідентифікація фізичних закономірностей;

– короткостроковий покроковий прогноз процесів і подій;

– довгостроковий покроковий прогноз;

– безмодельне прогнозування процесів за допомогою комплексування аналогів;

– апроксимація багатовимірних процесів;

– опис і прогнозування часових рядів;

– вибір обчислювальної схеми для вирішення диференціальних рівнянь;

– структурна ідентифікація об'єктів (модель вхід-вихід);

– класифікація / розпізнавання образів (пошук структур з вчителем);

– кластеризація / таксономія (пошук структур без вчителя);

– вибір структури моделі динамічного об’єкту [13];

– налаштування структури нейронних мереж (самоорганізація структури).

В приладобудуванні використовуються методи регресійного аналізу [14], методи планування експериментів [15], метод групового врахування експериментів [16, 17], метод групового врахування аргументів [5], математичні методи оптимізації, математичне програмування, вирішуються задачі апроксимації, екстраполяції та інтерполяції, задачі оптимізації методами покоординатного та градієнтного спуску тощо [18].

1.4. Вибір раціонального методу моделювання

Аналіз методів отримання математичних моделей при дослідженні параметрів процесу різання довів, що математичне моделювання досить широко та успішно використовується в приладобудуванні. Проте вибір раціонального методу моделювання залежить від конкретного об’єкту, який необхідно описати. Наприклад, якщо необхідно формалізовано описати структуру об’єкту, використовується статистичне моделювання, якщо процес, який виконує об’єкт, то створюється динамічна модель тощо. В кожному окремому випадку модель повинна задовольняти певним специфічним вимогам, а також основним для усіх моделей.

Вибір раціонального методу моделювання базується на наступних основних вимогах до моделі [19]:

– універсальність моделі є характеристикою, що відображає достовірність та ступінь повноти відповідності усіх властивостей вихідного об’єкту з отриманою моделлю;

– адекватність даної моделі, що є характеристикою, яка ілюструє відображення необхідних властивостей даного об’єкту з мінімальною похибкою або без неї;

– точність є характеристикою, яка повинна ілюструвати повне співпадіння всіх значень необхідних властивостей даного об’єкту та побудованої моделі;

– економічність – це характеристика, яка ілюструє, скільки часу та пам’яті комп’ютера необхідно витратити для повного опису моделі, її реалізації та введення в експлуатацію;

– складність моделі;

– потенційність моделі, або її прогнозованість з позиції отримання нових знань про досліджуваний об’єкт.

Даним вимогам повністю відповідають штучні нейронні мережі, що також є математичною моделлю. Перевагами штучних нейронних мереж, в порівнянні з багатьма методами математичного моделювання, є [20]:

– вирішення задач при невідомих залежностях. Використовуючи здатність навчання на багатьох прикладах, нейронна мережа може вирішувати задачі, в яких невідомі закономірності розвитку ситуації та залежності між вхідними та вихідними даними. Традиційні методи в даному випадку дадуть значно гірше описують систему;

– стійкість до шумів. Можливість роботи за наявності великого числа неінформативних, шумових вхідних сигналів. Немає необхідності робити їх попередній відсів, нейронна мережа сама визначить їх малопридатними для вирішення завдання і відкине їх;

– адаптованість до змін оточуючої середи. Нейронні мережі мають здатність адаптуватися до змін навколишнього середовища. Зокрема, нейронні мережі, що навчені діяти в певному середовищі, можуть бути легко перенавчені для роботи в умовах незначних коливань параметрів середовища. Більш того, для роботи в нестаціонарному середовищі (де статистика змінюється з плином часу) можуть бути створені нейронні мережі, що мають здатність до перенавчання в реальному часі;

– потенційно надвисока швидкодія. Нейронні мережі мають потенційно надвисоку швидкодію за рахунок використання масового паралелізму обробки інформації;

– стійкість до відмов при апаратній реалізації. Нейронні мережі потенційно стійкі до відмов. Це означає, що за несприятливих умов їх продуктивність падає незначно. Наприклад, якщо пошкоджено якийсь нейрон або його зв'язки, витяг запам’ятованої інформації ускладнюється. Однак, беручи до уваги розподілений характер зберігання інформації в нейронній мережі, можна стверджувати, що тільки серйозні пошкодження структури нейронної мережі істотно вплинуть на її працездатність. Тому зниження якості роботи нейронної мережі відбувається повільно;

– існування швидких алгоритмів. Нейронна мережа навіть при сотнях вхідних сигналів і десятках-сотнях тисяч еталонних ситуацій може бути швидко навчена на звичайному комп'ютері [21];

– можливість роботи зі скорельованими незалежними змінними, з різнотипною інформацією – неперервною та дискретною, кількісною та якісною, що зазвичай складно зробити застосовуючи методи статистики;

– мала кількість обмежень на структуру мережі та властивості нейронів. Тому за наявності експертних знань або у випадку спеціальних вимог можна цілеспрямовано обрати вид та властивості нейронів та самої мережі, збирати структуру нейронної мережі вручну, з окремих елементів, та задавати для кожного з них необхідні властивості.

Висновки до розділу і постановка задачі досліджень

Аналіз стану проектування та дослідження процесів і систем дозволив встановити, що основним методом їх реалізаціє є моделювання.

Огляд методів моделювання та їх класифікація, що виконано в даному розділі, встановив переваги і недоліки різних методів. Визначено, що найкращим методом моделювання є математичне моделювання.

В розділі розглянуто методи математичного моделювання. Встановлено, що найбільш ефективним і перспективним методом математичного моделювання процесів і систем є використання штучної нейронних мереж.

На основі наведеного аналізу літературних джерел стану моделювання і дослідження процесів і систем поставлено наступні задачі досліджень в дипломній роботі:

– розглянути основи штучних нейронних мереж та можливість їх використання для моделювання процесів і систем в технології приладобудування;

– виконати огляд засобів реалізації штучних нейронних мереж та вибір найбільш простого для використання;

– розглянути опис особливостей застосування платформи TensorFlow;

– виконати практичне застосування штучних нейронних мереж при розв'язанні різноманітних задач моделювання та прогнозування технологічних параметрів;

– розробити практичні рекомендації з використання штучних нейронних мереж за допомогою платформи TensorFlow для моделювання процесів і систем в технології приладобудування.

РОЗДІЛ 2.
ШТУЧНІ НЕЙРОННІ МЕРЕЖІ ДЛЯ МОДЕЛЮВАННЯ ПРОЦЕСІВ І СИСТЕМ

2.1. Сутність моделювання засобами штучних нейронних мереж

Сучасною математичною моделлю є штучна нейронна мережа, а також її програмна або апаратна реалізація, що побудована на принципі організації та функціонування біологічних нейронних мереж. Штучна нейронна мережа (ШНМ) являє собою систему з’єднаних та взаємодіючих між собою простих процесорів [22]. Застосування штучних нейронних мереж є прогресивним напрямком розвитку математичного моделювання. ШНМ можна використовувати для розпізнавання образів та класифікації, прийняття рішень та керування, кластеризації, прогнозування, апроксимації, стискання даних та асоціативна пам'ять, аналіз даних, оптимізація.

Штучна нейронна мережа є одним з основних інструментів в машинному навчанні. Нейронні мережі складаються з вхідного та вихідного шарів, а також прихованого шару, що складається з блоків, що використовують перетворення вхідних даних. Вони вирізняються відмінними інструментами для пошуку шаблонів, що надто важкі або чисельні для людини, аби обробити ці дані. Штучна нейронна мережа являє собою систему з’єднаних простих процесів (штучних нейронів), що взаємодіють між собою. Схему штучної нейронної мережі представлено на рисунку 2.1.

Існує декілька типів штучних нейронних мереж, кожен з яких використовується в певному випадку. Вони мають різні рівні складності.

Основним типом штучної нейронної мережі є штучна нейронна мережа прямого розповсюдження, де інформація від входу до виходу переміщується тільки в одному напрямку.

Найбільш використовуваним типом мережі є рекурентна штучна нейронна мережа. В ній данні можуть передаватися в декількох напрямках. Дані штучні нейронні мережі мають добрі здібності до навчання, тому широко застосовуються для вирішення задач типу розпізнавання мови або вивчення рукописного вводу.

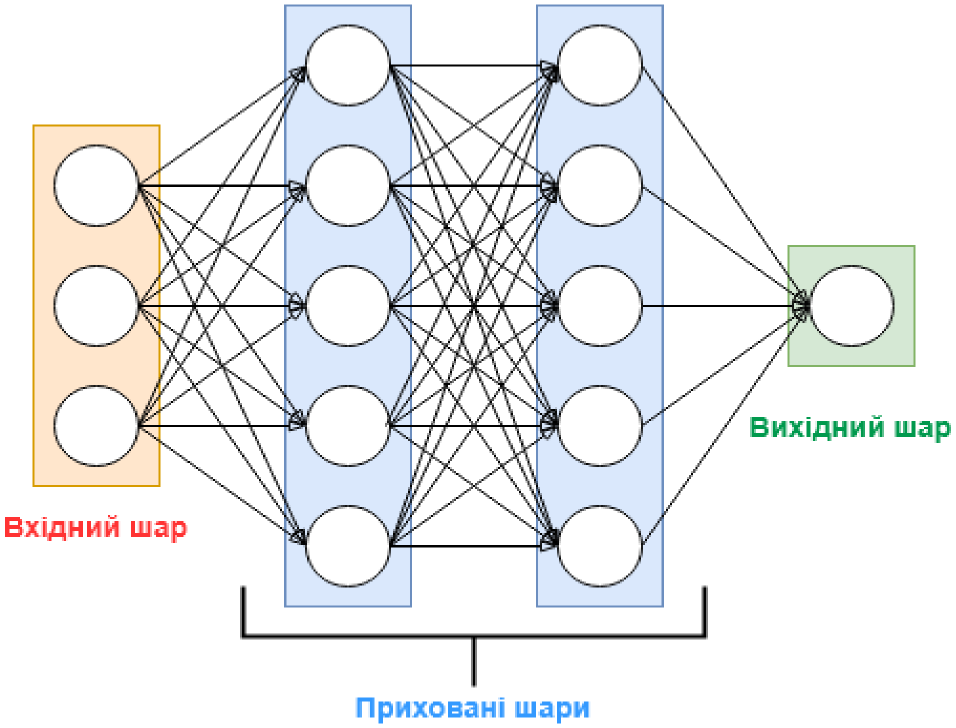


Рис. 2.1 Схема штучної нейронної мережі.

Також використовують згорткові штучні нейронні мережі, мережі Хопфілда тощо.

Вибір штучної нейронної мережі залежить від даних, на яких вона буде навчатися, поставленої задачі та багатьох інших факторів. В деяких випадках виникає необхідність використання декількох підходів до вирішення задачі, наприклад, розпізнавання голосу.

Штучні нейронні мережі, в цілому, використовуються для визначення закономірностей у даних. Задачі, які вирішує штучна нейронна мережа, можуть включати класифікацію, кластеризацію, прогнозування, апроксимацію, аналіз даних, оптимізацію, стиснення даних та асоціативну пам’ять. Класифікація наборів даних застосовується до попередньо визначених класів, кластеризація ж виконується за різними невизначеними категоріям, а прогнозування передбачає вгадування майбутніх подій на основі попередніх.

На технічному рівні однією з найбільших проблем штучних нейронних мереж є кількість часу, що необхідна для навчання штучної нейронної мережі, оскільки при розв’язанні більш складних задач вона може потребувати значного об’єму обчислювальних потужностей. Проте найбільшою проблемою є те, що штучні нейронні мережі представляють собою «чорні скрині», в які користувач вводить свої дані та отримує відповідь. Штучні нейронні мережі дають змогу точно налаштувати відповіді, проте вони не дають доступу до точного процесу прийняття рішень [23].

Моделювання за допомогою штучних нейронних мереж полягає в налаштуванні вхідного, вихідного, прихованих шарів, кількості нейронів та їх властивостей, що при проходженні певної кількість епох (кількість разів, що дані проходять крізь модель, для навчання) забезпечують необхідну точність.

За структурою зв’язків штучні нейронні мережі можуть бути згруповані в два основних класи:

– нейронні мережі прямого розповсюдження;

– рекурентні нейронні мережі.

В першому класі частіше використовуються багатошарові нейронні мережі, де штучні нейрони розташовані шарами, зв'язок між якими односпрямований і вихідні дані одного шару є вхідними для наступного. Такі мережі відносяться до статичних моделей, оскільки не мають ані зворотного зв’язку. ані динамічних елементів, а вихід не залежить від попередніх станів мережі [24]. Класифікація штучних нейронних мереж наведена на рис. 2.2.

Найпростішою одношаровою штучною нейронною мережею є Perceptron. Вона може мати багато входів, проте лише один вихід. На рисунку 2.3 для одного спостереження представлені різноманітні входи (незалежні змінні) $x_0, x_1, x_2, \ldots, x_n$. Кожен з цих входів множиться на вагу з’єднання або синапс, що представлені як $w_0, w_1, w_2, \ldots, w_n$. Вага ілюструє силу конкретного вузла.

Константа b на рисунку 2.3 представляє собою величину зміщення. Значення зміщення дозволяє зміщувати функцію вверх або вниз. В найбільш простому випадку ці добутки сумуються та передаються в передатну функцію (функцію активації) для генерації результату, що є вихідним. Математично це записується наступним чином:

$$x_0 * w_0 + x_1 * w_1 + x_2 * w_2 + \ldots + x_n * w_n = \sum_{i=0}^{n} (x_i * w_i) \qquad (2.1)$$

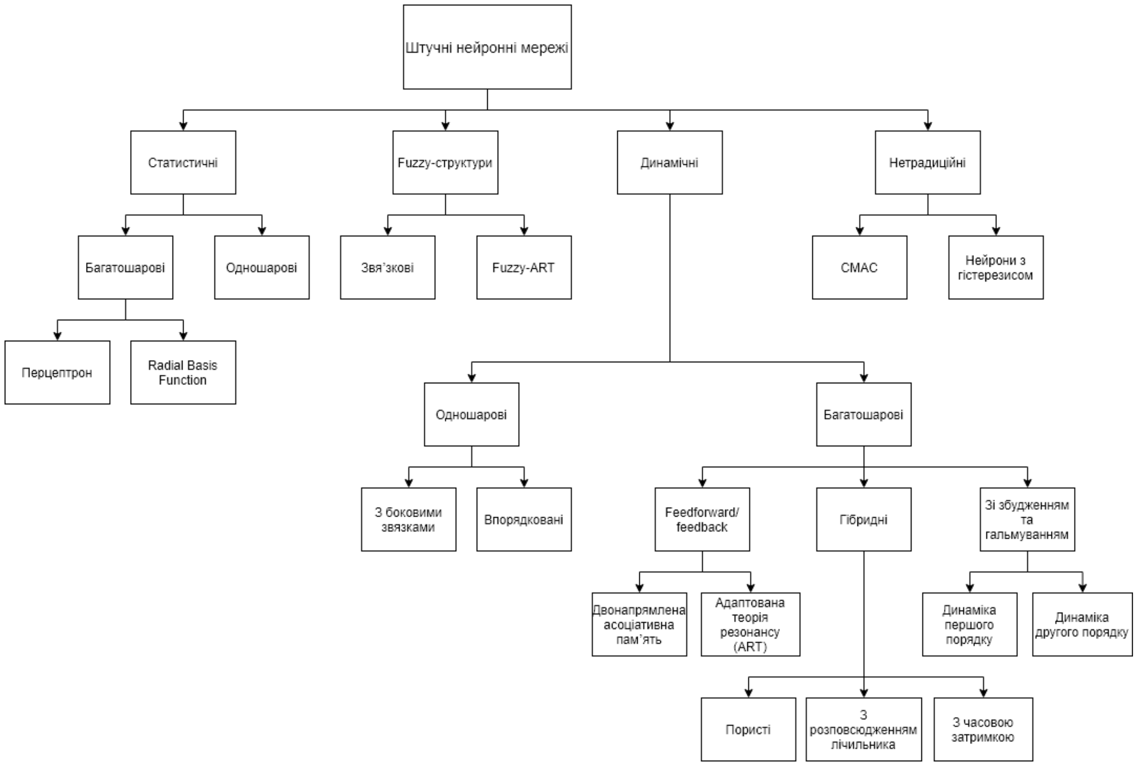


Рис.2.2. Класифікація штучних нейронних мереж.

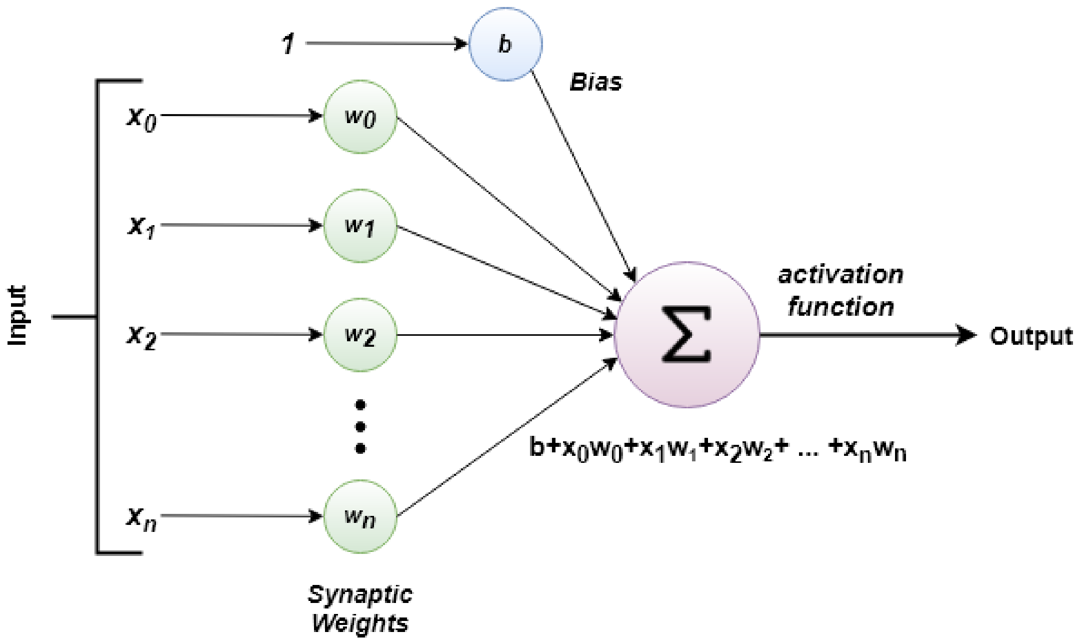


Рис. 2.3. Схема мережі Perceptron.

Такі моделі використовуються для вирішення різних складних задач:

– штучна нейронна мережа прямого розповсюдження – для задач регресії та класифікації.

– згорткові штучні нейронні мережі – для комп’ютерного зору.

– рекурентні штучні нейронні мережі – для аналізу часових рядів тощо.

Аби відобразити сутність біологічних нейронних систем нейрони отримують вхідні сигнали через декілька вхідних каналів, що мають певну інтенсивність (вагу), що відповідає синаптичній активності біологічного нейрону. З кожним нейроном обов’язково пов’язане якесь порогове значення, вираховується зважена сума входів, від якої необхідно відняти порогове значення, аби отримати пост-синаптичний потенціал нейрону (PSP) або величину активації нейрону. Даний сигнал активації перетворюється за допомогою функції активації (в теорії автоматизованого керування це називається передатною функцією) і в результаті отримується вихідний сигнал нейрону [25]. Аби підсилити потужність нейронної мережі та підвищити її точність можна додати прихований шар, що ускладнить модель, проте також може зменшити кількість витраченого часу на навчання.

Функція активації слугує для вирішення, чи є необхідність в активації нейрону, розраховуючи зважену суму та додаючи до неї зміщення. Ціллю є введення нелінійності у вихід нейрону. Якщо не використовувати функцію активації, то вихідний сигнал буде простою лінійною функцією, яка обмежена за своєю складністю та має меншу потужність. Також така модель не зможе вивчати та моделювати складні дані.

Навчання в нейронній мережі пов’язане з тим, як люди навчаються у своєму житті, тобто також базується на біологічних процесах. Штучні нейронні мережі можуть навчатися “із вчителем” для опису процесів, які перетворюють вхідні дані на вихідні, тобто для знаходження передатної функції. Навчання “з вчителем” передбачає, що для кожного вхідного вектору існує цільовий вектор, що являє собою необхідний вхід. Разом вони називаються “навчальною парою”. Зазвичай, ШНМ навчається на деякій кількості таких начальних пар. Схему навчання “з вчителем” представлено на рисунку 2.4.

Проте навчання “без вчителя” є більш правдоподібною моделлю навчання в біологічній системі. Даний метод навчання не потребує цільового вектору для виходів, а, отже, не передбачає порівняння з ідеальними значеннями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм налаштовує ваги так, аби було отримано вихідні вектори, тобто представлення досить близьких вхідних векторів давало однакові виходи. Процес навчання виділяє статистичні властивості навчальної множини та групує подібні вектори в класи.

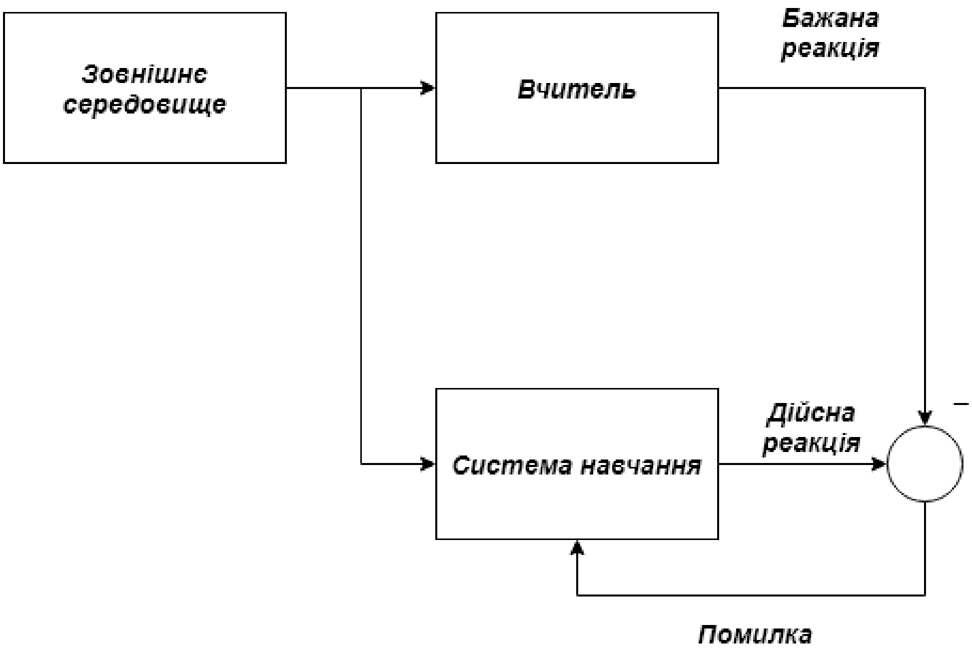


Рис. 2.4. Схема навчання “з вчителем”.

Подання на вхід вектору з даного класу дасть певний вихідний вектор, але до навчання неможливо передбачити, який саме вихід буде виконуватися даним класом вхідних векторів. Отже, виходи подібної мережі повинні трансформуватися в певну зрозумілу форму, що обумовлена процесом навчання. Проте це не є серйозною проблемою. Зазвичай, це не складно – визначити зв'язок між вхідними та вихідними даними, що встановила мережа (рис.2.5) [24].

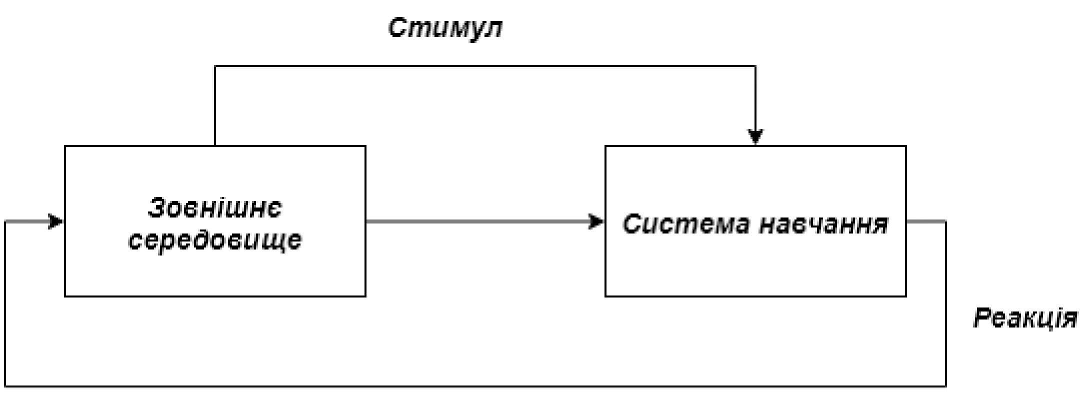


Рис.2.5. Схема навчання “без вчителя”.

На основі фактичних та прогнозованих значень вираховується функція втрат (loss function), як різниця між ними. Потім функція втрат надсилається назад у систему. Нашою метою є мінімізація функції втрат. Отже, доки існує різниця між фактичними та прогнозованими значеннями, є необхідність коригувати ваги, тобто повертати на вхід функцію втрат – створювати зворотній зв'язок. Цей процес необхідно повторювати, доки дана різниця не стане мінімально можливою. Дана процедура має назву зворотне розповсюдження та застосовується, поки значення loss function не стане мінімальним. Нейронна мережа, що керується даною процедурою, носить назву рекурентна нейронна мережа. Якщо в нейронній мережі дана процедура не використовується, то дану мережу називають нейронною мережею прямого розповсюдження.

Для одношарової нейронної мережі прямого розповсюдження для регулювання ваг найкраще підходить метод грубої сили. В даному випадку береться декілька можливих ваг, а далі намагаємося виключити інші ваги, окрім одного справа знизу U-подібної кривої. За допомогою простих методів виключення можна знайти оптимальні ваги, якщо є тільки одна вага. Проте даний метод має свої недоліки. Основним з них є те. що він не працює, якщо складну штучну нейронну мережу з багатьма вагами, що приведе до так званого «Прокляття розмірності» (Curse of Dimensionality).

Альтернативним методом, який підходить у випадку складної нейронної мережі є пакетно-градієнтний спуск (Batch-Gradient Descent). Даний алгоритм передбачає ітеративну оптимізацію першого порядку (рис. 2.6). Пакетно-градієнтний спуск передбачає мінімізацію loss function в процесі навчання з різними вагами або їх поновлення.

Перевага пакетно-градієнтного спуску в тому, що замість того, щоб працювати окремо з кожною вагою та відмічати невірні, необхідно дивитись на кут функціональної лінії. Якщо нахил негативний – це означає, що напрямок обрано донизу кривої, якщо ж позитивний – тоді не потрібно нічого робити. Таким чином проходить виключення великої кількості невірних ваг. Даний алгоритм працює добре, якщо крива випукла, в іншому випадку даний метод не підходить.

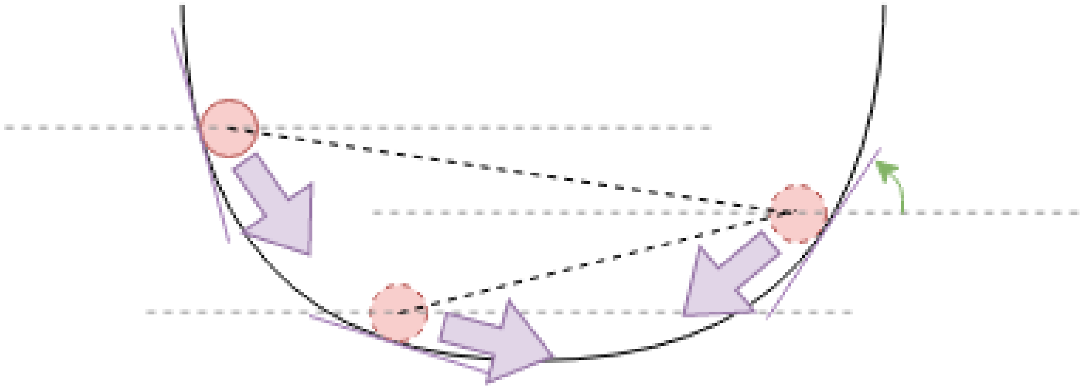


Рис.2.6. Схема градієнтного спуску.

Тому варто звернутися до одного з найпоширеніших методів – стохастично-градієнтного спуску (Stochastic Gradient Descent, далі – SGD) (рис. 2.7). В даному алгоритмі для кожної ітерації випадковим чином обирається один рядок за раз замість використання усього набору даних.

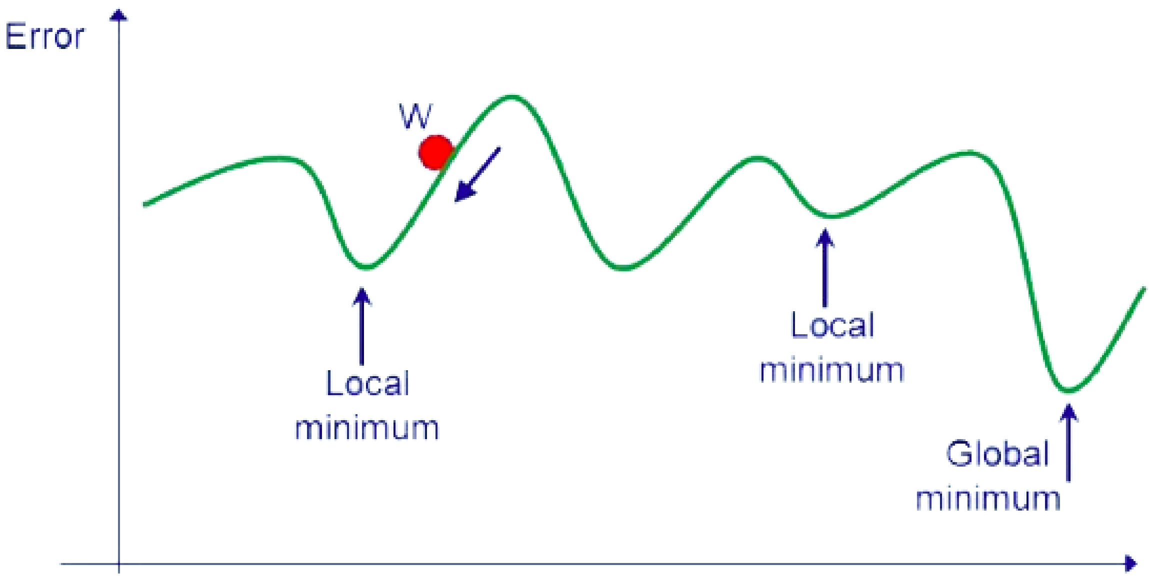


Рис.2.7. Схема статистично-градієнтного спуску.

В SGD береться один рядок даних, пропускається крізь нейронну мережу. а потім коригуються ваги. Для другого рядка робимо те саме, проте перед тим, як коригувати ваги, необхідно вирішити доцільність цього кроку порівнявши стару функцію втрат із тією, що було отримано.

Даний алгоритм дозволяє уникнути проблеми локальних мінімумів. Також SGD набагато швидше за Batch-Gradient Descent. Причиною цього є використання одного рядку за раз і не потребує завантаження усього набору даних в пам'ять для виконання обчислень, наслідком чого є необхідність у великій кількості ітерацій. Проте не дивлячись на велику кількість ітерацій, SGD потребує менших обчислюваних потужностей, аніж Batch-Gradient Descent. Отже, для оптимізації алгоритму навчання SGD має переваги перед Batch-Gradient Descent.

Тренування штучної нейронної мережі із застосуванням SGD виконується наступним чином [26]:

1. Довільно задаються ваги в межах від 0 до 1, проте не рівні 0.

2. Вводиться перший рядок даних у вхідний шар, кожний параметр в один вузол.

3. Пряме розповсюдження (Forward-Propagation) виконується зліва направо нейрони активуються таким чином, аби вплив активації кожного нейрону було обмежено вагами.

4. Порівняння фактичного результату із прогнозованим вимірюється loss function.

5. Зворотне розповсюдження (Back-Propagation) виконується справа наліво, помилка розповсюджується в зворотному напрямку. Оновлюються ваги, що найбільше виплинули на отримання похибки. В такому випадку швидкість навчання відповідає за кількість ваг, які необхідно оновити.

6. Повторюються кроки 1-5 і оновлюються ваги після кожного спостереження.

Проходження пунктів 1-6 називають епохою (epoch).

2.2. Області використання штучних нейронних мереж

Наразі штучні нейронні мережі посідають важливу роль і нашому житті. Області використання штучних нейронних мереж досить різноманітні: розпізнавання тексту або мови, розпізнавання емоцій, класифікація, кластеризація, прогнозування, семантичний пошук, експертні системи та системи підтримки прийняття рішень, прогнозування курсу акцій, системи безпеки, аналіз текстів [27]. Також штучні нейронні мережі застосовуються і в приладобудуванні [28-32].

У кожної предметної області при найближчому розгляді можна знайти постановки завдань для нейронних мереж. Ось список окремих областей, де рішення такого роду завдань має практичне значення вже зараз [33].

1. Економіка і бізнес: прогнозування часових рядів (курсів валют, цін на сировину, попиту, обсягів продажів), автоматичний трейдинг (торгівля на валютній, фондовій або товарній біржі), оцінка ризиків неповернення кредитів, передбачення банкрутств, оцінка вартості нерухомості, виявлення переоцінених і недооцінених компаній, рейтингування, оптимізація товарних і грошових потоків, зчитування і розпізнавання чеків і документів, безпеку транзакцій по пластикових картах.

2. Медицина і охорона здоров'я: постановка діагнозу хворому (діагностика захворювань), обробка медичних зображень, очищення показань приладів від шумів, моніторинг стану пацієнта, прогнозування результатів застосування різних методів лікування, аналіз ефективності проведеного лікування.

3. Авіоніка: автопілот, розпізнавання сигналів радарів, адаптивне пілотування сильно пошкодженого літака, безпілотні літальні апарати.

4. Зв'язок: стиснення відеоінформації, швидке кодування-декодування, оптимізація стільникових мереж і схем маршрутизації пакетів.

5. Інтернет: асоціативний пошук інформації, електронні секретарі та автономні агенти в інтернеті, фільтрація і блокування спаму, автоматична рубрикація повідомлень з стрічок новин, адресні реклама і маркетинг для електронної торгівлі, розпізнавання captcha.

6. Автоматизація виробництва: оптимізація режимів виробничого процесу, контроль якості продукції, моніторинг і візуалізація багатовимірної диспетчерської інформації, попередження аварійних ситуацій.

7. Робототехніка: розпізнавання сцени, об'єктів і перешкод перед роботом, прокладка маршруту руху, управління маніпуляторами, підтримання рівноваги.

8. Політологічні та соціологічні технології: передбачення результатів виборів, аналіз опитувань, прогноз динаміки рейтингів, виявлення значущих чинників, кластеризація електорату, дослідження і візуалізація соціальної динаміки населення.

9. Безпека, охоронні системи: розпізнавання осіб; ідентифікація особи за відбитками пальців, голосу, підписи або особі; розпізнавання автомобільних номерів, моніторинг інформаційних потоків в комп'ютерній мережі і виявлення вторгнень, виявлення підробок, аналіз даних з відеодатчиків і різноманітних сенсорів, аналіз аерокосмічних знімків.

10. Введення і обробка інформації: розпізнавання рукописних текстів, відсканованих поштових, платіжних, фінансових і бухгалтерських документів; розпізнавання голосових команд, голосове введення тексту в комп'ютер.

11. Геологорозвідка: аналіз сейсмічних даних, асоціативні методики пошуку корисних копалин, оцінка ресурсів родовищ.

З кожним роком штучні нейронні мережі все більше розвиваються, та на сьогодні найбільш активно використовуються в наступних областях [34]:

1. Машинне навчання засноване на навчанні штучного інтелекту на прикладі мільйонів подібних задач. Сьогодні машинне навчання активно впроваджується пошуковими системами Google, Yandex, Bing, Baidu. Таким чином, на основі мільйонів пошукових запитів, які ми кожний день вводимо в пошукові системи, їх алгоритми вчаться показувати нам найбільш відповідні результати, аби ми мали змогу знайти саме те, що нам потрібно.

2. Робототехніка. Тут нейронні мережі використовуються при розробці численних алгоритмів для так званих залізних «мізків» роботів.

3. Архітектори комп'ютерних систем використовують нейронні мережі для вирішення задачі паралельних (одночасних) обчислень.

4. За допомогою нейронних мереж математика може вирішувати різні складні математичні завдання.

5. В економіці. Тут нейронні мережі найчастіше використовуються для прогнозування цін, обмінних курсів, а також для оптимізації торгівлі на ринку. Найпопулярніший пакет нейронних мереж називається Brain Marker, його основна мета -

знайти рішення нетрадиційних завдань, таких як прогнозування акцій або моделювання різних ринкових ситуацій. Він заснований на нейронної мережі, яка вчиться на багатьох прикладах.

6. В системах безпеки і охорони: розпізнавання осіб. Технологія розпізнавання осіб з використанням нейронної мережі дозволяє захищати мобільні пристрої від доступу інших осіб і, відповідно, від крадіжки особистих даних, таких як Apple Face ID, а також допомагає знаходити й ідентифікувати злочинців в місцях, де велика кількість людей переміщується за допомогою відеокамер, встановлених там - штучна нейронна мережа МВС.

7. У промисловому виробництві. Як приклади успішного використання нейронних мереж в цій області можна згадати оптимізацію режимів виробничого процесу, управління процесами обробки, запобігання аварійним ситуаціям тощо. Великий досвід був накопичений в використанні штучних нейронних мереж в управлінні якістю в промисловості. Наприклад, нейронна мережа, яка використовується на підприємствах Intel для виявлення дефектів у виробництві мікросхем, здатна бракувати несправний чіп з точністю до 99,5%. А фахівці Національного інституту стандартів і технологій (NIST), застосовуючи звукові хвилі і отримуючи відбитий сигнал, а потім обробляючи результат за допомогою штучної нейронної мережі, перевіряють якість бетону при товщині матеріалу до півметра.

8. У медицині. Штучні нейронні мережі знайшли широке застосування для вирішення завдань медичної діагностики. Наприклад, група вчених зі Стенфордського університету розробила нейронну мережу, яка може діагностувати пневмонію на основі декількох рентгенівських променів, і робить це не гірше, ніж практикуючі рентгенологи. Такі програми допоможуть діагностувати захворювання у пацієнтів у віддалених регіонах, де відчувається нестача кваліфікованих кадрів.

9. Значно збільшилася кількість військових розробок з використанням нейронних мереж, орієнтованих на створення надшвидких, «розумних» суперкомп'ютерів.

10. В геології нейронні мережі прямого розповсюдження є найбільш перспективними для вирішення завдань картографування ландшафтно-екологічних систем. Завдання картування території полягає в тому, щоб визначити характеристики

яскравості фізіогномічних особливостей - рослинності, відкритих ділянок ґрунту або водних поверхонь, аналізуючи супутникові зображення за допомогою нейронної мережі.

11. В інтернет-технологіях. Тут на основі нейронних мереж впроваджуються такі технології, як електронні секретарі і призначені для користувача агенти в інтернеті, фільтрація інформації, блокування спаму і автоматична рубрикація новинних каналів.

2.3. Використання штучних нейронних мереж в технології приладобудування

Штучні нейронні мережі та штучний інтелект досить широко використовується багатьох сферах життєдіяльності. Проте важко буде знайти таку область використання, яка буде використовувати більше переваг, аніж виробничий сектор. великі компанії світу активно вкладають кошти у рішення для машинного навчання у своїх виробничих процесах та досягають вражаючих результатів. Від зниження витрат на робочу силу до підвищення продуктивності праці та загальної швидкості виробництва, штучний інтелект – за допомогою також промислового інтернету речей (Internet of Things, далі – IoT) – вступає в еру інтелектуального виробництва. Наразі є наступні приклади поточних реалізацій [35]:

1. Загальне покращення процесу. Рішення на основі машинного навчання можуть обслуговувати щоденні процеси протягом всього виробничого циклу. Використовуючи дану технологію, виробники можуть виявляти різноманітні проблеми в своїх звичних методах виробництва, від вузьких місць до нерентабельних ліній. За допомогою комбінації машинного навчання та промислового IoT компанії глибше вивчають питання керування логістикою, запасами, активами та цілями поставок, що дає цінну інформацію, що розкриває потенційні можливості не тільки у процесі виробництва, але і в процесах пакування та дистрибуції. Добрим прикладом є німецька компанія-конгломерат Siemens, що використовує нейронні мережі для моніторингу своїх металургійних заводів у пошуках потенційних проблем, що можуть вплинути на ефективність їх праці. За допомогою комбінації датчиків, що встановлено на обладнанні, та власного інтелектуальної хмари компанія здатна контролювати, записувати та аналізувати кожний крок, пов'язаний з виробничим процесом.

2. Розробка продукту. Одним з найбільш розповсюджених застосувань машинного навчання є етап розробки продукту. Це пов'язано з тим, що стадії проектування та планування нових продуктів, а також покращення існуючих пов'язане з великою кількістю інформації, яку необхідно враховувати для отримання найкращих результатів. Таким чином, машинне навчання допомагає у збиранні даних про споживачів та їх аналізі для розуміння потреб, виявлення прихованих потреб та виявлення нових бізнес-можливостей. Все це в кінцевому випадку призводить до появи більш якісної продукції з існуючого каталогу, а також нової продукції, яка може відкрити нові джерела прибутку для компанії. Машинне навчання особливо добре допомагає знизити ризики, пов'язані з розробкою нової продукції, оскільки знання, які воно надає, забезпечують етап планування для прийняття більш обґрунтованих рішень. CocaCola, один з найбільших брендів у світі. використовує машинне навчання для розробки продуктів. Фактично, запуск у виробництво Cherry Sprite був результатом використання компанією машинного навчання. Компанія використовувала інтерактивні диспансери з газованими напоями, де покупці мали змогу додавати різні аромати до базових напоїв свого каталогу. CocaCola зібрала отримані дані та використала машинне навчання для виявлення найбільш частих комбінацій. В результаті вони виявили досить великий ринок для надання нового напою усій країні.

3. Контроль якості. За правильного використання машинне навчання може покращити якість кінцевого продукту до 35% [36], особливо в окремих галузях обробної промисловості. Є два способи, якими машинне навчання може це зробити. В першу чергу виявлення аномалій в товарах та пакуванні. Шляхом глибокого вивчення продукції, що виробляється, компанії можуть запобігти потраплянню бракованої продукції на ринок. Насправді, є дослідження, які говорять про покращення процесу виявлення дефектів на 90% в порівнянні з людськими перевірками [37]. Також глибокі нейронні мережі можуть допомогти в доступності, продуктивності, якості збірного обладнання та слабких сторонах машини. Siemens використовує нейронні мережі для моніторингу виробництва сталі та підвищення загальної продуктивності [38].

Наступним способом є можливе покращення якості виробничого процесу. За допомогою пристроїв IoT та додатків машинного навчання компанії можуть аналізувати доступність та продуктивність всього обладнання, що використовується у виробничому процесі. Це дозволяє визначити найкращий час для проведення профілактичного обслуговування конкретного обладнання, аби подовжити строк його служби та уникнути вартісних простоїв у виробництві. General Electric є одним з найбільших інвесторів у відділ контролю якості, особливо в усе, що пов'язане з профілактичним обслуговуванням. Дана компанія вже створила та запровадила свої інструменти на базу машинного навчання в більше, ніж сотні тисяч активах у своїх підрозділах та у клієнтів, включаючи аерокосмічну, енергетичну та транспортну галузі. Його системи працюють для виявлення ранніх попереджуючих ознак аномалій на своїх виробничих лініях та забезпечення прогнозу з довготривалими оцінками поведінки та життя.

4. Безпека. Оскільки всі ці рішення для машинного навчання засновано на додатках, операційних системах, мережах хмарних та локальних платформах, безпека використовуваних мобільних додатків, пристроїв та даних є обов'язковою для сучасного виробника. На щастя, машинне навчання має відповідь у рамках Zero Trust Security (ZTS) [39]. Завдяки цій технології доступ користувачів до цінного цифрового доступу та інформації строго регулюється та обмежується. Таким чином, машинне навчання може використовуватися для аналізу того, як окремі користувачі отримують доступ до певної захищеної інформації, які використовують додатки та як вони до неї під'єднуються. Позначаючи строгий периметр навколо цифрових активів, машинне навчання може визначати. хто отримує доступ до інформації, а хто – ні.

5. Роботи. В решті решт, деякі роботи стають розумнішими за допомогою машинного навчання. Використання штучного інтелекту в роботах дозволяє їм виконувати рутинні завдання, які є складними або небезпечними для людини. Ці нові роботи перевершують лінії, на яких вони колись були зібрані, оскільки можливості машинного навчання дозволяють їм вирішувати більш складні процеси, ніж раніше. Саме на це націлена німецька виробнича компанія KUKA зі своїми промисловими роботами. Його ціллю є створення роботів, які можуть працювати разом з людьми та діяти як їх працівники. Дані роботи оснащені високопродуктивними датчиками, які дозволяють їм виконувати складні завдання. працюючи поряд з людьми, та пізнавати, як підвищити свою продуктивність. Сама KUKA використовує своїх роботів на власних

заводах, проте є й інші великі виробники, які так само виробляють роботів. Наприклад, BMW, відома автомобільна марка, є одним з її найбільших клієнтів та одним з підприємств, що вже впевнилися в тому, що роботи можуть знизити кількість помилок. пов’язаних з людської працею, підвищити продуктивність та цінність усього виробничого ланцюгу. Також є японський виробник промислової робототехніки та технологій автоматизації Fanuc. Дана компанія використовує глибоке навчання з підкріпленням (тип рішення для машинного навчання), що дозволяє її роботам швидко та ефективно навчати себе новим навичками без необхідності точного та складного програмування [38].

6. Керування ланцюгами надходжень. Машинне навчання допомагає максимізувати цінність компанії, покращуючи її логістичний процес, керуючи запасами, активами та ланцюгами надходжень. Ця комбінація та інтеграція пристроїв машинного навчання, штучного інтелекту та IoT допомагає забезпечити високу якість даних процесів. Сучасні виробники шукають способи поєднати нові технології з відслідковуванням активів, точністю, видимістю ланцюга постачань та оптимізацією запасів. Компанії-розробники машинного навчання розробили пакет управління ланцюгом постачань. який контролює кожний етап виробництва. пакування та доставки [38].

Отже, завдяки штучним нейронним мережам обробна промисловість наразі є технічно розвиненою галуззю. Протягом десятиліть виробники були першими, хто запровадив усі види технологій від автоматизації до складних цифрових рішень та робототехніки. Це призводить до того, що виробники усього світу вкладають кошти у рішення для машинного навчання з метою розширення можливостей своїх виробництв. Результатами впровадження штучних нейронних мереж, машинного навчання та штучного інтелекту є підвищення продуктивності, впровадження вдосконалених продуктів, покращення розподілу, зменшення кількості браку та скорочення кількості відмов обладнання – і це тільки деякі з усіх переваг. І хоча наразі дані рішення далекі від широкого розповсюдження, увага багатьох компаній сконцентрована на більш розумному способі виробництва продукції.

2.4. Засоби моделювання, що реалізують штучні нейронні мережі

На даний час розроблено велику кількість програмних засобів та пакетів, що реалізують штучні нейронні мережі. Вони підтримують різні мови програмування. мають різні можливості, різний функціонал та інструментарій. Програмне забезпечення для штучних нейронних мереж використовується для моделювання, дослідження, розробки та застосування штучних нейронних мереж, програмних концепцій, адаптованих до біологічних нейронних мереж. Програмне забезпечення штучних нейронних мереж призначене для практичного застосування штучних нейронних мереж з упором на інтелектуальний аналіз даних та прогнозування. Ці симулятори аналізу даних зазвичай мають деякі можливості попередньої обробки та використовують відносно просту статистичну нейронну мережу, яку можливо налаштувати [40].

1. Neural Designer – це десктопний додаток для інтелектуального аналізу даних, в якому використовуються штучні нейронні мережі – головна парадигма машинного навчання. Особливостями є високопродуктивні розрахунки; проста у використанні візуалізація та аналіз даних високого рівня.

2. Neuroph – це полегшений фреймворк нейронних мереж на мові програмування Java для розробки загальних архітектур нейронних мереж. Він містить добре розроблену бібліотеку Java з відкритим вихідним кодом. Neuroph спрощує розробку нейронних мереж, представляючи бібліотеку та інструмент GUI, який підтримує створення, навчання та збереження нейронних мереж. Особливостями є проста у використанні структура; підтримка розпізнавання тексту; нормалізація даних; підтримка розпізнавання зображень; шаблон прогнозу фондового ринку.

3. Darknet – це інфраструктура нейронної мережі з відкритим вихідним кодом. що написано на C та CUDA. він швидкий, простий в установці та підтримує розрахунки на базі центрального процесору (CPU) та графічного (GPU). Більш прийнятне для використовування на Linux та Mac операційних системах. Особливостями є виявлення об’єктів в реальному часі; класифікація ImageNet;

Nightmare; рекурентні нейронні мережі Darknet; DarkGo; Tiny Darknet; навчання класифікатору на CIFAR-10.

4. TensorFlow- це комплексна платформа з відкритим вихідним кодом для машинного навчання. Має всеосяжну та гнучку екосистему інструментів. бібліотек та ресурсів, що дозволяє використовувати найсучасніші технології машинного навчання, а також легко створювати та розгортувати додатки на основі машинного навчання. Особливостями є проста побудова моделі; надійна розробка машинного навчання будь-де; потужні експерименти для досліджень [41].

5. Keras – це бібліотека глибокого навчання для Theano та TensorFlow. Це високорівнева бібліотека нейронних мереж. що написано на Python та здатна працювати поверх Theano або TensorFlow. Його було розроблено на основі TensorFlow з метою забезпечення швидкого експериментування. Здатність переходити від ідеї до результату з найменш можливою затримкою є ключем до проведення добрих дослідів. Бібліотека глибокого навчання Keras дозволяє легко та швидко створювати прототипи (завдяки повній модульності, мінімалізму та розширюваності). Вона підтримує як згорткові нейронні мережі. так і рекурентні нейронні мережі, а також їх комбінації. Також має широку документацію та інструкції для розробників. Особливостями є модульність; мінімалізм; легка розширюваність та робота з Python.

6. NeuroSolutions – це простий у використанні пакет програм для нейронних мереж, що розроблено під операційну систему Windows. Його розроблено для того, щоб зробити нейронні мережі простими та доступними, як для починаючих, так і для досвідчених розробників. Аналіз нейронної мережі складається з трьох основних етапів: навчання нейронної мережі на основі даних, тестування нейронної мережі на точність та прогнозування / класифікацію на основі нових даних. Особливостями є кластерний аналіз; прогнозування продажів; спортивні прогнози та медична класифікація.

7. Tflearn – це модульна та прозора бібліотека глибокого навчання, що побудована на TensorFlow. Його було розроблено для представлення високорівневого API для TensorFlow, щоб полегшити та пришвидшити

експерименти, при цьому залишаючись повністю прозорим та сумісним з ним. Високорівневий API в даний час підтримує більшість останніх моделей глибокого навчання. таких як Convolutions, LSTM, BiRNN, BatchNorm, PReLU, Residual networks, Generative networks. Особливостями є проста та красива візуалізація графіків з докладною інформацією про ваги. градієнти, активацію тощо; потужні допоміжні функції для навчання будь-якого графу TensorFlow з підтримкою декількох входів, виходів, оптимізаторів.

8. ConvNetJS – це бібліотека JavaScript для навчання моделей глибокого навчання (нейронних мереж) виключно в браузерах користувачів. Немає вимог до програмного забезпечення, немає компіляторів, немає установок, немає GPU. Особливостями є загальні модулі нейронної мережі; функції втрат класифікації та регресії; можливість створювати та навчати згорткові нейронні мережі, що оброблюють зображення.

9. Torch – це наукове середовище для розрахунків з широкою підтримкою алгоритмів машинного навчання. Він пропонує користувачам просту у використанні та ефективну програму завдяки простій та швидкій скриптовій мові LuaJIT та базовій реалізації на C / CUDA. Особливостями є швидка та ефективна підтримка GPU; процедури лінійної алгебри та числової оптимізації.

10. NVIDIA DIGITS спрощує задачі глибокого навчання, такі як управління даними, проектування та навчання нейронних мереж на системах з декількома GPU, моніторинг продуктивності в режимі реального часу за допомогою сучасних візуалізацій та вибір найбільш ефективної моделі у браузері результатів для розгортування. Особливостями є моніторинг продуктивності в режимі реального часу; розширені візуалізації; Amazon Machine Image.

11. DeepPy – ліцензоване середовище глибокого навчання MIT. DeepPy дозволяє програмувати на Python на основі NumPy. Особливостями є програмування на Python; штучні нейронні мережі прямого розповсюдження.

12. Knet – це середовище глибокого навчання, яке підтримує роботу з GPU та автоматичне диференціювання з використанням динамічних розрахункових графів. Використовує динамічні обчислювальні графи, згенеровані під час виконання, для

автоматичної диференціації майже будь-якого коду Julia. Особливостями є лінійна регресія, класифікація Softmax, багатошаровий перцептрон, згорткова нейронна мережа та рекурентна нейронна мережа.

13. Microsoft Cognitive Toolkit - це комерційний інструментарій з відкритим вихідним кодом, що дозволяє використовувати інтелектуальні можливості у великих наборах даних за допомогою глибокого навчання, забезпечуючи безкомпромісне масштабування, швидкість і точність з комерційною якістю і сумісністю з вже використовуваними мовами програмування і алгоритмами [42].

14. Accord.NET Framework - це інфраструктура машинного навчання .NET, об'єднана з бібліотеками обробки звуку і зображень, повністю написаними на C#, це платформа для створення виробничих програм комп'ютерного зору, комп'ютерного прослуховування, обробки сигналів і статистики навіть для комерційного використання.

15. NeuroIntelligence - це додаток для нейронних мереж, призначений для допомоги експертам з нейронних мереж, інтелектуального аналізу даних, розпізнавання образів і прогнозного моделювання в рішенні реальних задач. NeuroIntelligence використовує тільки перевірені алгоритми моделювання нейронних мереж і методи нейронних мереж; програмне забезпечення швидке і просте у використанні.

16. Zebra by Mipsology - це ідеальний комп'ютерний движок глибокого навчання для виведення нейронних мереж. Zebra легко замінює або доповнює CPU / GPU, що дозволяє будь-якій нейронній мережі виконувати обчислення швидше, з меншим енергоспоживанням і з меншими витратами.

17. Neural Network Toolbox – це пакет розширення MATLAB, що містить інструменти для проектування, моделювання, розробки та візуалізації нейронних мереж. Надає алгоритми, попередньо навчені моделі і програми для створення, навчання, візуалізації і моделювання нейронних мереж з одним прихованим шаром (званим дрібної нейронною мережею) і нейронних мереж з декількома прихованими шарами (так званими глибокими нейронними мережами). Завдяки використанню запропонованих інструментів можна виконувати класифікацію, регресію,

кластеризацію, зменшення розмірності, прогнозування часових рядів, а також динамічне моделювання і управління системою [43].

2.5. Вибір раціональних методів і засобів штучних нейронних мереж

На основі аналізу програмних засобів для моделювання штучних нейронних мереж було обрано платформу TensorFlow з бібліотекою Keras. Основними параметрами для вибору програмних засобів було:

– безкоштовне програмне забезпечення;

– мова програмування: Python або C++;

– зрозумілий та зручний інтерфейс;

– доступність у будь-який час;

– велика кількість додаткових інструментів для навчання;

– можливість пошарового створення багатошарової штучної нейронної мережі прямого розповсюдження для прогнозування параметрів;

– можливість збереження готового рішення (навченої штучної нейронної мережі);

– можливість збереження тільки найкращого рішення з усіх.

Дані вимоги повністю задовольняє платформа TensorFlow з бібліотекою Keras. Для компіляції коду та запуску коду було обрано Google Colaboratory, що є також повністю доступним, безкоштовним, зрозумілим та підтримує мову програмування Python. Також він зберігає код на Google Drive. що надає доступ до програмного коду у будь-який час та на будь-якому пристрої (девайсі), де є Інтернет з’єднання. Google Colaboratory дає можливість обирати процесор для виконання програми: GPU та CPU.

Також найбільшою перевагою TensorFlow для розвитку машинного навчання є абстракція. Замість того, аби розбиратися з найдрібнішими подробицями реалізації алгоритмів або вишукувати правильні способи прив'язки виведення однієї функції до введення іншої, розробник може зосередитися на загальній логіці програми. TensorFlow піклується про деталі за лаштунками.

TensorFlow пропонує додаткові зручності для розробників, яким необхідно налагоджувати і аналізувати додатки TensorFlow. Режим “Eager execution” дозволяє оцінювати та змінювати кожну операцію графу окремо та прозоро, замість того, щоб будувати весь граф у вигляді одного непрозорого об'єкта і оцінювання його відразу [44].

Висновки до розділу

В даному розділі встановлено, що сучасним методом математичного моделювання є використання штучних нейронних мереж, які дозволяють вирішувати величезну кількість різноманітних задач, як наприклад, прогнозування, класифікація, кластеризація, прогнозування часових рядів тощо. Вони знайшли своє застосування у багатьох сферах людської праці, таких як економіка і бізнес, медицина і охорона здоров'я, авіоніка, зв'язок та Інтернет, автоматизація виробництва, безпека та захист інформації, введення і обробка інформації, геологорозвідка, робототехніка тощо.

Особливо широке використання штучні нейронні мережі отримали у промисловому виробництві. Вони допомагають підвищити продуктивність, впровадити більш досконалу продукцію, покращення логістики, зменшення кількості браку та скорочення кількості відмов обладнання.

Оскільки штучні нейронні мережі вже добре зарекомендували себе у приладобудуванні, то для моделювання ТП та прогнозування їх параметрів було обрано саме ШНМ.

Аналіз програмних засобів та інструментарію показав, що усім поставленим вимогам відповідають платформа TensorFlow разом з бібліотекою Keras.

РОЗДІЛ 3.
ПЛАТФОРМА TENSORFLOW І ОСОБЛИВОСТІ ЇЇ ВИКОРИСТАННЯ ДЛЯ МОДЕЛЮВАННЯ ПРОЦЕСІВ І СИСТЕМ

3.1. Області використання TensorFlow

TensorFlow – це гнучка та багатофункціональна платформа з відкритим вихідним кодом для машинного навчання. Дана платформа має гнучку та всеосяжну екосистему інструментів та бібліотек., що дозволяє використовувати найновіші та найсучасніші засоби машинного навчання (Machine Learning), а також легко створювати додатки на основі машинного навчання.

TensorFlow було розроблено дослідниками та інженерами, що працюють в команді Google Brain в межах дослідницької організації машинного інтелекту Google для проведення машинного навчання та досліджень глибоких нейронних мереж. Система досить загальна, аби її можна було застосовувати в інших областях [45]. Використовується для задач побудови та тренування штучної нейронної мережі для автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття. TensorFlow може працювати на багатьох паралельних процесорах, таких як CPU та GPU. Обчислення TensorFlow представляють потік даних через граф станів. Назва TensorFlow йде від операцій з багатовимірними масивами даних, які також називають “тензорами” [46].

TensorFlow надає стабільні API-інтерфейси Python, але також існують реалізації для C#, C++, Haskel, Java, Go та Swift.

TensorFlow застосовується для вирішення різноманітних задач, таких як:

– задачі регресії;

– задачі класифікації;

– задачі кластеризації;

– задачі комп’ютерного зору;

– задачі прогнозування;

– задачі розпізнавання;

– усі задачі, які можуть бути вирішені за допомогою штучних нейронних мереж.

Для кращих результатів та швидшого моделювання нейронних мереж використовується Keras – високорівневий фреймворк, що працює поверх TensorFlow, що дозволяє вирішувати велику кількість рутинних задач, таких як, наприклад, перетворення тексту в числові послідовності для аналізу або швидке моделювання штучних нейронних мереж для машинного навчання. Також Keras має вже вбудовані моделі такі, як Лінійна або поліноміальна регресія [47].

Основні приклади використання TensorFlow [48]:

1. Розпізнавання голосу/звуку. Одним з найбільш відомих прикладів використання TensorFlow є додатки на основі звуку. За правильному надходженні даних нейронні мережі здатні розуміти аудіосигнали. Це можуть бути:

– розпізнавання голосу – в основному використовується у IoT, автомобілебудуванні, безпеці та UX / UI;

– голосовий пошук - в основному використовується в телекомунікаціях, у виробників телефонів;

– аналіз настроїв - в основному використовується у CRM;

– детектоскопія (шум двигуна) – в основному використовується в автомобілебудуванні та авіації.

Що стосується поширених випадків використання, усім знайомий голосовий пошук і голосові помічники у нових широко поширених смартфонах, такі як Siri від Apple, Google Now для Android і Microsoft Cortana для Windows Phone.

Розуміння мови є ще одним поширеним варіантом використання для розпізнавання голосу. Додатки перетворення мови в текст можуть використовуватися для визначення фрагментів звуку у великих аудіофайлах та «переписування» сказаного слова у вигляді тексту.

Звукові додатки також можуть бути використані в CRM. Сценарій варіанту використання може бути наступним: алгоритми TensorFlow замінюють агентів обслуговування клієнтів та направляють клієнтів до необхідної інформації, до того ж швидше, аніж агенти.

2. Текстові додатки. Ще одним популярним застосуванням TensorFlow є текстові додатки, такі як сентиментальний аналіз (CRM, соціальні мережі), виявлення загроз (соціальні мережі, уряд) та виявлення шахрайства (страхування, фінанси). Визначення мови – одне з найпопулярніших застосувань текстових додатків:

– усім відомий Google Translate, який підтримує переклад понад 100 мов з однієї на іншу. Розвинені версії можуть використовуватися в багатьох особливих випадках, таких як переклад жаргонної мови в контрактах на зрозумілу мову;

– додавання тексту. Google також виявив, що для більш коротких текстів можна узагальнювати за допомогою методики, так званим послідовним навчанням. Це може бути використано для створення заголовків для новинних статей;

– інший варіант використання Google – SmartReply. Він автоматично генерує відповіді електронною поштою.

3. Розпізнавання зображень. В основному використовується виробниками соціальних мереж, телекомунікацій та мобільних телефонів. Розпізнавання лиця, пошук зображень, виявлення руху, машинний зір і кластеризація фотографій можуть використовуватися також в автомобільній, авіаційній та медичній галузях. Розпізнавання зображень має на меті розпізнавати та ідентифікувати людей та об'єкти на зображеннях, а також розуміти зміст і контекст зображення.

Алгоритми розпізнавання об'єктів TensorFlow класифікують та ідентифікують довільні об'єкти на великих зображеннях. Це зазвичай використовується в інженерних додатках для визначення фігур для цілей моделювання (побудова тривимірного простору з 2D-зображень) і в соціальних мережах для позначки фотографій (Deep Face від Facebook). Наприклад, аналізуючи тисячі фотографій дерев, технологія може навчитися ідентифікувати дерево, яке ніколи не бачило раніше.

Розпізнавання зображень починає розширюватися і в сфері охорони здоров'я, де алгоритми TensorFlow можуть обробляти більше інформації і виявляти більше шаблонів, ніж їх людські аналоги. Комп'ютери тепер можуть переглядати скановані зображення і виявляти більше хвороб, ніж люди.

4. Часові ряди. Алгоритми часових рядів TensorFlow використовуються для аналізу даних часових рядів з метою вилучення необхідної або значимої статистики.

Вони дозволяють прогнозувати невизначені періоди часу на додаток до створення альтернативних версій часових рядів.

Найбільш поширеним варіантом використання часових рядів є рекомендація (Recommendation). Це використання є поширеним у Amazon, Google, Facebook і Netflix, де вони аналізують активність клієнтів і порівнюють її з мільйонами інших користувачів, щоб визначити, що клієнт хотів би купити або подивитися. Ці рекомендації стають ще розумнішими. Наприклад, вони пропонують певні речі в якості подарунків або телешоу, які можуть сподобатися членам вашої родини.

RankBrain розроблено Google, це великомасштабне розгортання глибоких нейронних мереж для ранжирування пошуку на www.google.com. Це частина алгоритму пошуку, який використовується для сортування мільярдів сторінок, про які він знає, і знаходить ті, які вважає найбільш актуальними.

Інше використання алгоритмів часового ряду TensorFlow в основному представляє інтерес для фінансів, бухгалтерського обліку, державного управління, безпеки та IoT з виявленням ризиків, прогнозним аналізом і плануванням підприємства / ресурсів.

5. Виявлення рухів. Нейронні мережі TensorFlow також працюють з відеоданими. Робота з відеоданими в основному полягає у виявленні руху, виявленні потоків в реальному часі в іграх, безпеці, аеропортах та UX / UI. Останнім часом університети працюють над великомасштабними наборами даних класифікації відео, такими як YouTube-8M, з метою прискорення досліджень в області великомасштабного розуміння відео, навчання представленню, моделювання зашумлених даних, навчання переносу та адаптації доменів для відео.

В наслідок того, що TensorFlow є бібліотекою з відкритим вихідним кодом, скоро побачимо безліч інноваційних варіантів використання, які будуть впливати один на одного і робити внесок в технологію машинного навчання.

3.2. Продукти, що створено з використанням TensorFlow

Існує також ряд мобільних додатків, які демонструють використання бібліотеки машинного навчання.

Деякі інші хороші продукти, створені з використанням TensorFlow [49]:

1. Google Creative Labs Teachable Machine, яка використовує TensorFlow.js, що дозволяє вам навчати штучну нейронну мережу за допомогою камери вашого комп'ютера, в реальному часі в браузері.

2. Nsynth Super – розроблений Google Creative Labs, який дозволяє створювати музику з використанням абсолютно нових звуків, що генеруються алгоритмом Nsynth.

3. Giorgio Camthat – дозволяє створювати музику, натискаючи на картинки.

Крім того, NASA розробляє систему з TensorFlow для класифікації орбіт і кластеризації об'єктів астероїдів і буде класифікувати і прогнозувати навколоземні об'єкти.

Ця бібліотека безумовно прискорює навчання, надаючи інструменти, які завжди були відсутні.

3.3. Структура платформи TensorFlow

TensorFlow має велику кількість програмних модулів, класів та функцій, які використовуються для різних задач: класифікація, кластеризація, прогнозування, розпізнавання текстів/образів тощо. Усі модулі можна подивитися на сайті з офіційною документацією TensorFlow [50]. Проте, виділимо ті модулі, що є необхідними для прогнозування параметрів:

– initializers – це модуль, що визначає спосіб задання випадкових ваг шарам штучної нейронної мережі;

– keras – реалізація API Keras;

– losses – вбудовані функції втрат, що використовуються для оцінки моделі під час навчання. Під час навчання дану функцію намагаються мінімізувати;

– metrics – вбудовані функції, що використовуються для оцінки продуктивності моделі. Схоже до функцій втрат, проте не використовуються для навчання мережі;

– optimizers – вбудовані функції оптимізаторів, які поновлюють вагові параметри, аби мінімізувати функцію втрат;

– saved_model – це операція TensorFlow, що необхідні для збереження моделей та наступного їх відтворення без оригінальної моделі;

– summary – операції для запису зведених даних для використання в аналізі та візуалізації;

– version – операція для визначення наявної версії TensorFlow.

Keras як надбудова TensorFlow також має велику кількість модулів, класів та функцій для створення нейронних мереж. Ось основні з них:

– initializers – це модуль, що визначає спосіб задання випадкових ваг шарам штучної нейронної мережі;

– regularizers – дозволяє застосовувати штрафи до параметрів шару або активності шару під час оптимізації. Ці штрафи сумуються у функції втрат, що оптимізують мережу;

– activations – функції активації шарів мережі;

– losses– вбудовані функції втрат, що використовуються для оцінки моделі під час навчання. Під час навчання дану функцію намагаються мінімізувати;

– metrics – вбудовані функції, що використовуються для оцінки продуктивності моделі. Схоже до функцій втрат, проте не використовуються для навчання мережі;

– optimizers – вбудовані функції оптимізаторів, які поновлюють вагові параметри, аби мінімізувати функцію втрат;

– callback – це об’єкт, який може виконувати дії на різних етапах навчання (на початку або в кінці епохи, до або після одного пакету);

– sequential – це стек шарів для створення багатошарової нейронної мережі прямого розповсюдження.

Як видно, деякі програмні моделі повторюються і це нормально. Їх можна використовувати і як модулі TensorFlow, і як модулі Keras, оскільки вони рівноцінні. Проте треба враховувати, що результат функцій metrics буде у вигляді тензору, тому краще застосовувати бібліотеку sklearn для використання функцій metrics для оцінки моделі.

3.4. Алгоритм моделювання засобами TensorFlow

Для правильного застосовування нейронних мереж, необхідно розробити алгоритм моделювання засобами TensorFlow. Пропонується наступний алгоритм моделювання засобами TensorFlow для створення нейронної мережі прямого розповсюдження (багатошарового перцептрону):

1. Необхідно імпортувати необхідні бібліотеки;
2. Підготовка даних;
3. Створення моделі прямого розповсюдження:
– додавання шарів до моделі;
– налаштування шарів моделі;
– задання розмірності вхідних даних;
– задання кількості нейронів в шарі;
– задання функції активації;
– задання функції ініціалізації;
4. Налаштування функції компіляції:
– вибір оптимізатора та налаштування;
– вибір функції втрат;
– задання run_eagerly = True;
5. Налаштування функції тренування:
– вказання вхідних та вихідних даних для тренування;
– задання кількості епох;
– налаштування функції зворотного виклику;
– вказання валідаційних даних (не використовуються для навчання);
6. Тренування нейронної мережі.
7. Прогнозування тестових даних.
8. Вибір методу оцінки моделі.
9. Запуск написаного коду та оцінка результатів.
10. Внесення змін до налаштувань.
11. Збереження моделі.

12. Використання моделі.

Даний алгоритм використовується для розв'язання задач регресії. Рекурентні нейронні мережі зазвичай застосовують для розпізнавання образів, обробки голосу, прогнозування введення наступної літери тощо, оскільки дані задачі досить складні, тож потребують більш складного та багатокомпонентного розв'язку. Тому в даній роботі рекурентні нейронні мережі не описуються.

Розглянемо кожний пункт окремо.

1. Імпорт необхідних бібліотек та модулів. Імпорт модулів та бібліотек повністю залежить від функцій, що будуть використані, а вже в процесі програмування буде поповнюватися. Основними спочатку є наступні бібліотеки та модулі:

– tensorflow – для підключення безпосередньо TensorFlow;

– numpy – бібліотека мови Python, що необхідна для багатовимірних масивів та матриць;

– sklearn – це бібліотека мови Python, в якій знаходиться велика кількість алгоритмів для задач, пов'язаних з класифікацією та машинним навчанням, а також функції, необхідні для оцінки результатів.

2. Підготовка даних. Даний пункт означає, що нам необхідно або імпортувати дані, або вручну їх записати у код як масив даних за допомогою бібліотеки numpy. Також можна їх підготувати за допомогою нормалізації або масштабування. Keras має вбудовані функції для цього. Розберемо, як вони працюють.

Нормалізація. Це перетворення даних від вхідного інтервалу [min; max] до інтервалу [0;1]. Значення нормалізуються наступним чином:

$$y = \frac{x - \min}{\max - \min}, \tag{3.1}$$

де x – це один з елементів вхідного інтервалу;

max – це максимальне значення вхідного інтервалу;

min – це мінімальне значення вхідного інтервалу.

Доброю практикою є використання об'єкту MinMaxScaler з бібліотеки scikit-learn:

– налаштування MinMaxScaler застосувавши дані для навчання. Для нормалізації це означає, що дані для тренування будуть використані для знаходження min та max значення даних. Це можна зробити, викликавши функцію fit().

– застосування шкали до тренувальних даних. Це означає, що можна використати шкалу для нормалізації даних, а потім навчити моделі. Це виконується викликом функції transform().

– застосування шкали для зворотного перетворення. Аби отримати не нормалізований результат, необхідно виконати зворотне перетворення за допомогою функції inverse_transform().

– застосування шкали для нових даних. Так само, як і застосування до тренувальних даних, можна використовувати дану шкалу для нормалізації тестових або валідаційних даних. Проте треба враховувати, що нові дані не повинні виходити за інтервал [min; max] тренувальних даних, оскільки в такому випадку втрачається сенс використання нормалізації даних (інтервал не буде рівним [0;1].

Стандартизація. Також Keras дає можливість стандартизувати дані. Стандартизація передбачає зміну масштабу розподілу даних таким чином, аби середнє значення спостережуваних значень було рівне 0, а стандартне відхилення – 1. Як і нормалізація, стандартизація необхідна, якщо вхідні дані мають різний масштаб. Стандартизацію краще виконувати, якщо дані відповідають гаусівському закону, інакше – можна отримати ненадійні результати. Стандартизація вимагає, аби можна було точно оцінити середнє та стандартне відхилення спостережуваних значень.

Значення стандартизуються наступним чином:

$$y = \frac{x - mean}{st\_dev}, \tag{3.2}$$

де mean – це середнє значення усіх значень x з вхідного діапазону;

st_dev – це стандартне відхилення.

Середнє значення розраховується за формулою:

$$mean = \frac{\sum_{i=0}^{n} x_n}{n}, \tag{3.3}$$

де n – кількість значень x.

Стандартне відхилення розраховується за наступною формулою:

$$st\_dev = \sqrt{\frac{\sum_{i=0}^{n}(x - mean)^2}{n}}, \tag{3.4}$$

Оцінки середнього та стандартного відхилення набору даних можуть бути стійкішими до нових даних, аніж min та max.

Доброю практикою для масштабування даних є використання об’єкту StandartScaler з бібліотеки scikit-learn. Алгоритм використання StandartScaler є таким самим, як і MinMaxScaler.

3. Створення моделі прямого розповсюдження. В даному пункті використовується клас Sequential, що групує лінійний стек шарів у tf.keras.Model (батьківський клас). Sequential забезпечує навчання та вивід функцій у даній моделі.

Sequential модель підходить для простого стеку шарів, де кожний шар має рівно один вхідний тензор та рівно один вихідний. Вона не підходить, коли:

– модель має декілька входів або декілька виходів;

– будь-який шар має декілька входів або виходів;

– необхідно зробити загальний доступ до шарів;

– необхідна нелінійна топологія.

Sequential має свої методи:

– add(). Sequential.add(layer) додає екземпляр шару поверх стеку шарів. Даний метод може приймати один основний аргумент: layer, що є екземпляром рівня.

– pop(). Sequential.pop() видаляє останній шар в моделі.

Створення послідовної моделі можна виконати двома рівноцінними способами [51]:

– передавши список шарів конструктору Sequential.

– також можна послідовно створювати послідовну модель за допомогою add() методу.

Наступним кроком є вибір шарів і їх налаштування.

Є декілька типів вбудованих шарів:

Input object – шар вхідного об’єкту, що зазвичай використовується для вказання кількості вхідних даних. В якості аргументу приймає тільки розмірність вхідних даних. Використовується для створення тензору Keras [52].

Activation layer – шар активації, застосовує функцію активації до виходу. В якості аргументу приймає значення функції активації [53].

Embedding layer – перетворює натуральні числа (індекси) в щільні вектори фіксованого розміру [54].

Masking layer – маскує послідовність, використовуючи значення маски для пропускання часових кроків [55].

Lambda layer – забезпечує довільні вирази як Layer об’єкт. Даний шар використовується таким чином, що довільні функції TensorFlow можуть використовуватися при побудови Sequential та Functional API моделей. Дані шари краще за все підходять для простих операцій або швидких експериментів. Для більш складних варіантів використання застосовуються підкласи tf.keras.layers.Layer [56].

Dense layer – щільно зв’язаний шар нейронної мережі. Він приймає наступні аргументи [57]:

– units: додатне ціле число, розмірність вихідної кількості даних або ще кажуть, що це кількість нейронів у шарі.

– input_shape: Альтернатива Input object.

– activation: функція активації. За замовчуванням використовується лінійна активація.

– use_bias: Boolean, вказує, чи використовує даний шар функцію зміщення.

– kernel_initializer: Ініціалізатор для матриці ваг.

– bias_initializer: Ініціалізатор для вектору зміщення.

– kernel_regularizer: Функція регуляризації, що застосовується до матриці ваг.

– bias_regularizer: Функція регуляризації, що застосовується до вектору зміщення.

– activity_regularizer: Функція регуляризації, що застосовується до вихідного шару (його «активація»). Альтернатива Activation layer.

– kernel_constraint: Функція обмеження, що застосовується до матриці ваг.

– bias_constraint: Функція обмеження, що застосовується до вектору зміщення.

Отже, основним шаром, що входить до моделі Sequential та який ми будемо налаштовувати, є Dense layer. Повний опис усіх аргументів, а також значення, яких вони можуть набувати, можна подивитися в офіційній документації [58-61].

Далі розглянемо основні параметри Dense layer більш докладно.

Типи функцій регуляризації [62]:

– Data Augmentation – використовується для маркування даних. Даний метод генерує нові дані навчання з вихідного набору даних. Це також дешевий та простий спосіб збільшення об’єму даних для тренування.

– Dropout – це техніка, за якої випадково обрані нейрони ігноруються під час тренування. Приклад використання Dropout приведено на рисунку 3.1.

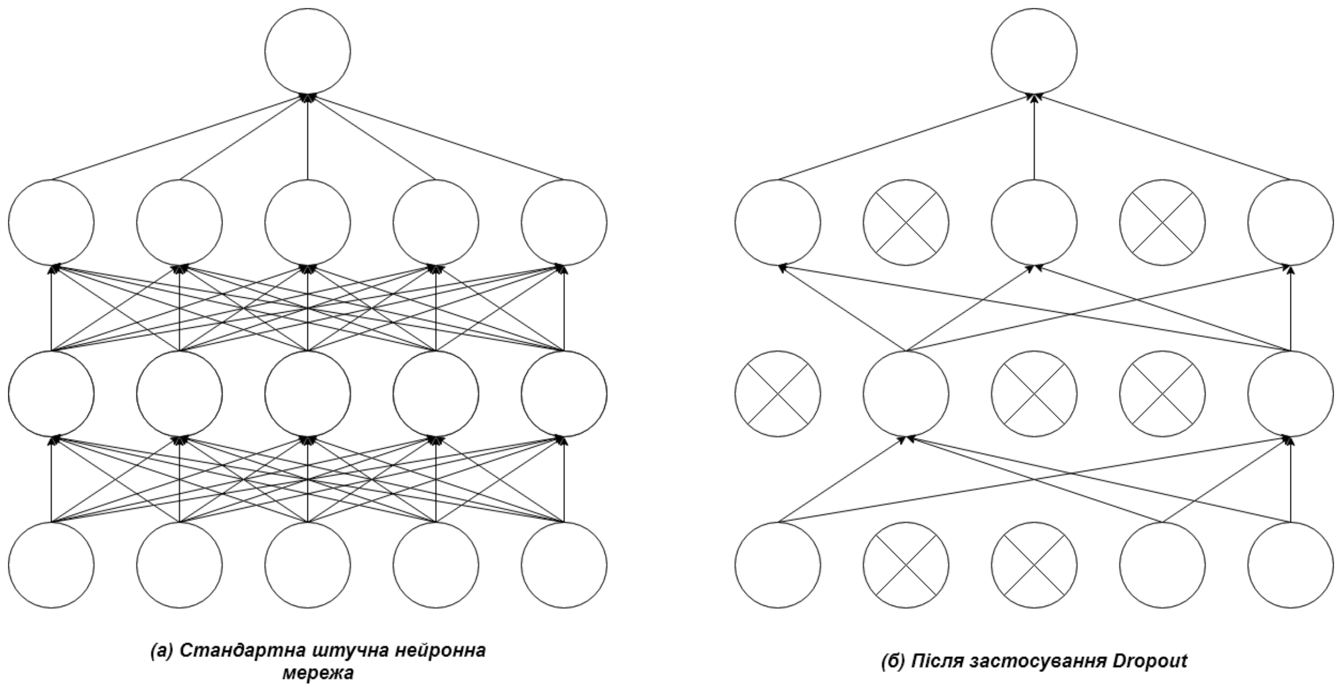


Рис. 3.1. Штучна нейронна мережа із застосуванням Dropout регуляризації. (а): Стандартна нейронна мережа із двома прихованими шарами.
(б): Приклад розрідженої штучної нейронної мережі, що є результатом застосування функції регуляризації Dropout до штучної нейронної мережі (а). Закреслені нейрони було виключено.

– Early Stopping використовується задля запобігання перенавчання на тренувальному наборі даних та поганих результатів на тестовому. Задача полягає в тому, щоб навчати мережу достатньо для того, щоб вона могла вивчати відображення від входів до виходів, але не навчати модель так довго, аби вона не відповідала навчальним даним. При побудові моделі вона оцінюється на наборі валідаційних даних після кожної епохи. Якщо точність моделі на валідаційному або тренувальному наборі починає погіршуватися (t), тоді процес навчання припиняється (рис. 3.2) [62].

– Ensembling включає в себе декілька методів машинного навчання в одну модель для прогнозування. Найбільш розповсюдженими є Bagging та Boosting. Bagging використовує складні базові моделі та намагається «згладити» їх прогнози, в той час коли Boosting використовує прості базові моделі та намагається підвищити їх сукупну складність.

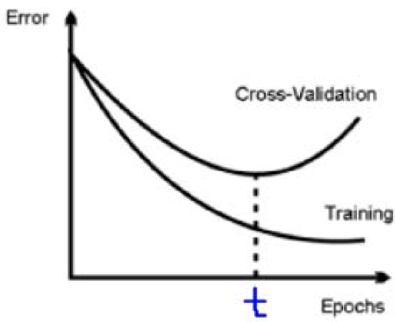


Рис. 3.2. Приклад залежності величини помилки від кількості епох на валідаційних та тренувальних даних.

– Injecting Noise – це метод, що включає в себе зашумлення даних для навчання. Шум часто вводять у вхідні дані для розширення набору даних. Одним з підходів до покращення помилки узагальнення та структури задачі відображення є додавання випадкового шуму. Це означає, що мережа менше здатна запам’ятовувати навчальні вибірки, оскільки вони постійно змінюються, що призводить до менших ваг мережі та більш стійкій мережі з меншою похибкою узагальнення. Шум додається тільки під час тренування. Під час оцінки та при використанні моделі для прогнозування нових даних шум не додається.

– L1 Regularization – це модель регресії, в якій використовується метод L1 Regularization називається метод Ласо. Вона є моделлю вибору, коли кількість функцій велика, оскільки вона представляє собою розріджені рішення. Можна отримати обчислювальні переваги за рахунок ігнорування функцій з нульовими коефіцієнтами.

– L2 Regularization – це модель регресії, в якій використовується метод регуляризації L2 Regularization називається регресійна система хребта. Основна різниця між L1 Regularization та L2 Regularization полягає в тому, що L2 Regularization використовує квадратичну величину коефіцієнту в якості штрафного члену функції втрат. L2 Regularization може вирішити проблему мультиколінеарності, обмежуючи норму коефіцієнта та зберігаючи усі змінні. Також може використовуватися для оцінки важливості предикторів та покарання предикторів, які не важливі.

Типи функцій активації [63]:

– Linear. Лінійна функція, де активація пропорційна входу, який є зваженою сумою від нейрону. Математично це записується наступним чином:

$$R(z,m) = \{z * m\}, \tag{3.5}$$

де z – вхідне значення;

m – коефіцієнт функції активації.

Також, у вигляді коду можна представити наступним чином:

```
def linear (z, m):
return m*z
```

На рисунку 3.3 зображено графік лінійної функції при m = 1 [63].

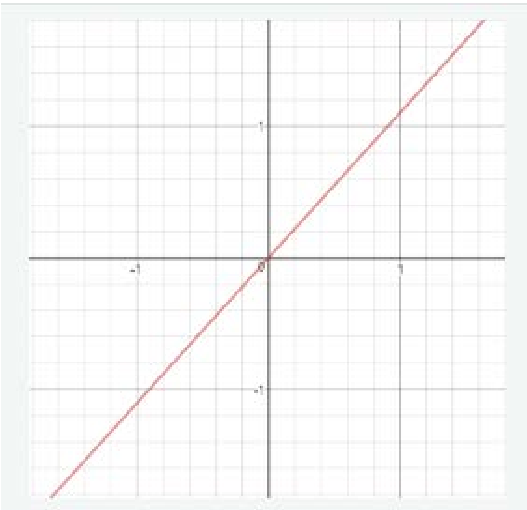


Рис 3.3. Графік лінійної функції.

– ELU. Функція, яка, як правило, швидше зводить витрати до нуля та дає більш точні результати. На відміну від інших функцій активації, ELU має допоміжну альфа-константу, яка повинна бути додатнім числом. Математично це записується наступним чином:

$$R(z) = \begin{cases} z, & z > 0 \\ \alpha * (e^z - 1), & z \leq 0 \end{cases}, \tag{3.6}$$

де α – додатна константа;

z – вхідне значення.

У вигляді програмного коду функція ELU представляється наступним чином:

```
def elu(z, alpha):
return z if z >= 0 else alpha*(e^z – 1)
```

Графік функції ELU представлено на рисунку 3.4 [63].

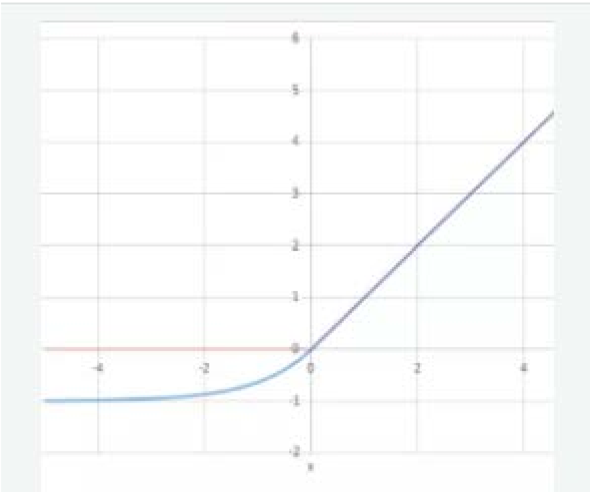


Рис. 3.4. Графік функції ELU.

– ReLU. Функція активації, яка на відміну від ELU, яка повільно згладжується до тих пір, поки її вихід не стане рівним – α, згладжується досить різко. Математично дана функція має наступний вигляд:

$$R(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}, \tag{3.7}$$

де z – вхідне значення.

У вигляді програмного коду дана функція має наступний вигляд:

```
def relu(z):
return max(0, z)
```

На рисунку 3.5 зображено графік функції ReLU [63].

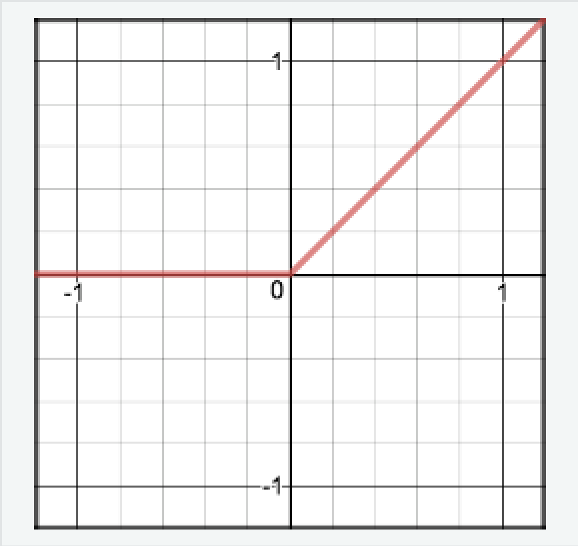


Рис. 3.5. Графік функції ReLU.

– LeakyReLU. LeakyRelu – це варіант ReLU. Замість того, щоб бути рівним 0 при z < 0 неповний ReLU допускає ненульовий градієнт α (зазвичай, α = 0.01). Математично дана функція записується наступним чином:

$$R(z) = \begin{cases} z, & z > 0 \\ \alpha z, & z \leq 0, \end{cases} \tag{3.8}$$

де α – додатна константа;

z – вхідне значення.

Програмно функція LeakyReLU записується наступним чином:

```
def leakyrelu(z, alpha):
return max(alpha*z, z)
```

На рисунку 3.6 зображено графік функції LeakyReLU [63].

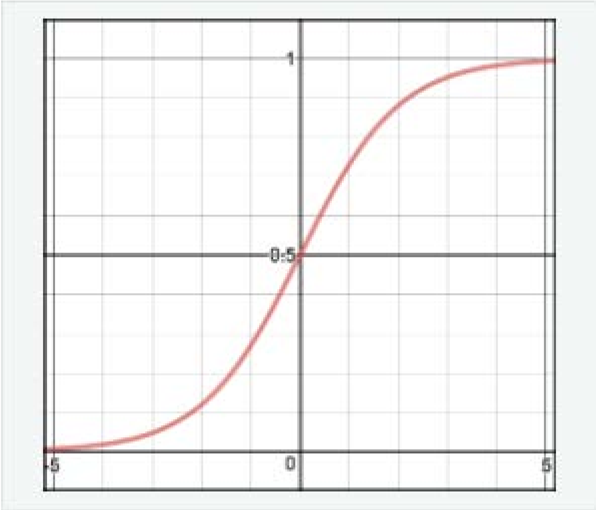


Рис. 3.6. Графік функції LeakyReLU.

– Sigmoid. Приймає реальне значення в якості вхідного та виводить інше в діапазоні від 0 до 1. Функція Sigmoid записується наступним чином:

$$S(z) = \frac{1}{1+e^{-z}}, \tag{3.9}$$

де z – вхідне значення.

Програмно дана функція записується наступним чином:

```
def sigmoid(z):
return 1.0 / (1+np.exp(-z))
```

Графік даної функції зображено на рисунку 3.7 [62].

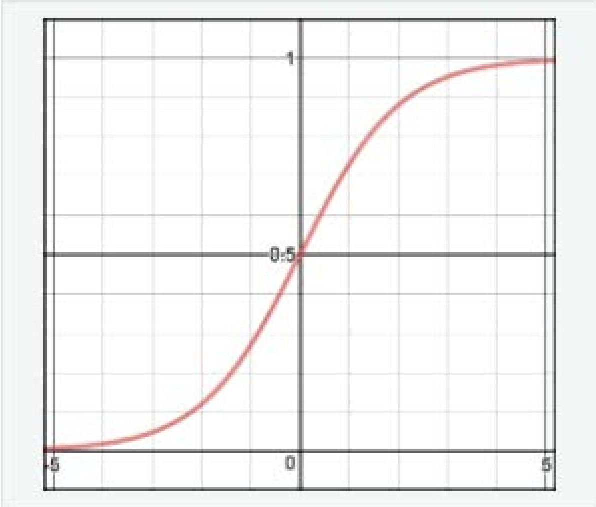


Рис. 3.7. Графік функції Sigmoid.

– Softmax. Дана функція розраховується розподілом ймовірностей події n відповідно до інших подій. В загальному, дана функція обчислює ймовірності кожного цільового класу за усіма цільовими класами.

– Tanh. Стискає дійсне число до діапазону [-1; 1]. Дана функція нелінійна. проте на відміну від Sigmoid, його продуктивність орієнтована на нуль. Тож, на практиці дана функція є більш задовільною, аніж Sigmoid. Математично функція Tanh має наступний вигляд:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \tag{3.10}$$

де z – вхідне значення.

Програмно ж вона записується наступним чином:

```
def tanh(z):
return (np.exp(z) – np.exp(-z) / np.exp(z) + np.exp(-z))
```

На рисунку 3.8 зображено графік функції Tanh [63].

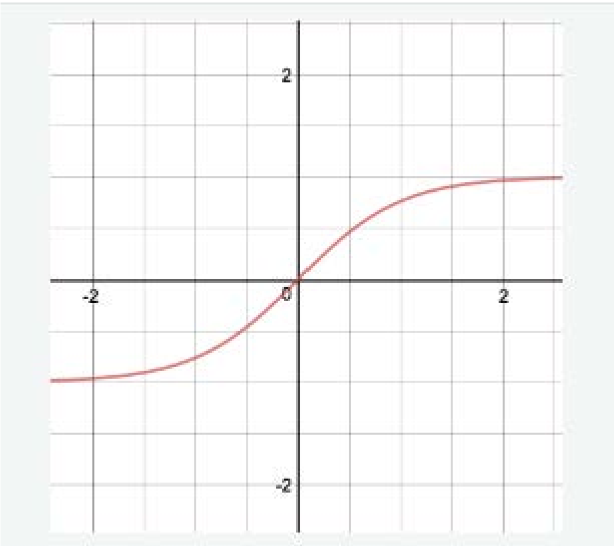


Рис. 3.8. Графік функції Tanh.

Функції ініціалізації [58]:

– Constant – ініціалізатор, що генерує тензори з постійними значеннями;

– GlorotNormal – ініціалізатор Glorot Normal, що також має назву звичайний ініціалізатор Ксав’є;

– GlorotUniform – ініціалізатор Glorot Uniform, що також має назву універсальний ініціалізатор Ксав’є;

– Identity – ініціалізатор, що генерує ідентичну матрицю;

– Ones – ініціалізатор, що генерує тензори, ініціалізовані в 1;

– Orthogonal – ініціалізатор, що генерує ортогональну матрицю;

– RandomNormal – ініціалізатор, що генерує тензори з нормальним розподілом;

– RandomUniform – ініціалізатор, що генерує тензори з рівномірним розподілом;

– TruncatedNormal – ініціалізатор, що генерує усічений нормальний розподіл;

– VarianceScaling – ініціалізатор, що здатен адаптувати свою шкалу за формою тензорів ваг;

– Zeros – ініціалізатор, що генерує тензори, ініціалізовані в 0.

4. Налаштування функції компіляції. Функція compile() приймає наступні основні аргументи:

optimizer: Функція оптимізації. Офіційна документація за посиланням [64]. Типи функцій оптимізації [65]:

– Adagrad – встановлює швидкість навчання відповідно до параметру;

– Adadelta – є продовженням Adagrad, проте, на відміну від нього, замість накопичування усіх минулих квадратів градієнтів, Adadelta обмежує вікно накопичуваних минулих градієнтів до фіксованого розміру ваг [66];

– Adam – розраховує адаптивні швидкості навчання для кожного параметру;

– Conjugate Gradients – розраховує квадратну задачу оптимізації за кінцеве число кроків [67];

– BFGS – на відміну від ньютонівських методів не потребує розрахунку повного розрахунку гесіана функції, тобто немає необхідності знаходити часткові похідні другого порядку [68];

– Momentum – враховує минулі градієнти, аби згладити оновлення;

– Nesterov Momentum – має більшу вірогідність збіжності для випуклих функцій, тому на практиці працює іноді краще, ніж звичайний Momentum [69];

– Newton's Method – має квадратичну швидкість збіжності, тож є ще кращим, аніж Nesterov Momentum. проте потребує більшої обчислювальної потужності [70];

– RMSProp – працює, зберігаючи експоненційно зважене середнє квадратів минулих градієнтів;

– SGD – стохастичний градієнтний спуск.

loss: Функція втрат. Офіційна документація за посиланням: [71].

metrics: Список показників, які будуть оцінюватися моделлю під час навчання та тестування. Список таких показників за посиланням: [72].

Також може приймати аргументи loss_weights, sample_weight_mode, weighted_metrics.

5. Налаштування функції тренування. Функція fit() приймає основні аргументи [73]:

– x – вхідні дані, які можуть бути представлені у вигляді:

а) масив даних Numpy або список масивів (якщо модель має декілька входів);

б) тензор TensorFlow або список тензорів;

в) словникове відображення вхідних імен у відповідний масив/тензор, якщо модель має іменовані входи;

г) набір даних tf.data;

д) генератор або keras.utils.Sequence.

– y – цільові дані. Як і вхідні дані x, це може бути або масив Numpy, або тензор TensorFlow. Тип вихідних даних повинен відповідати типу вхідним даним;

– batch_size – може бути цілий натуральним числом або None (використовується, якщо немає необхідності використовувати даний параметр, або розмір даних невеликий). Це кількість зразків для поновлення градієнту. За замовчуванням приймає значення 32;

– epochs – ціле натуральне число. Кількість епох для навчання моделі;

– verbose – 0, 1 або 2. Режим багатомовності. 0 = тиша, 1 = індикатор виконання, 2 = один рядок за епоху.;

– callbacks – список keras.callbacks.Callback екземплярів. Список зворотних викликів, що використовуються під час навчання;

– validation_data – дані, за якими можна оцінити втрати, та будь-які метрики моделі в кінці кожної епохи. Модель не буде навчатися на цих даних;

– run_eagerly – це параметр, який повинен завжди бути True, аби мережа навчалася «з нетерпінням» (eagerly).

6. Тренування нейронної мережі. Даний пункт передбачає тренування штучної нейронної мережі за допомогою функції fit().

7. Прогнозування тестових даних. Прогнозування даних виконується за допомогою функції predict(). Вона генерує вихідні прогнози відповідно до вхідних вибірок. Основні аргументи, яких набуває дана функція [73]:

– x – вхідні дані;

– batch_size – Кількість зразків на партію;

– verbose – режим багатомовності (0 або 1);

– callbacks – список keras.callbacks.Callback екземплярів. Список зворотних викликів, що використовуються під час навчання.

8. Вибір методу оцінки моделі. Визначити похибку можна за допомогою вбудованої функції evaluate(), що повертає значення втрат та метрики для моделі. Також можна використати бібліотеку sklearn, що має наступні функції [74]:

– max_error() – метрика, що визначає максимальну залишкову помилку;

– coverage_error() – це міра похибки покриття;

– mean_absolute_error() – середня абсолютна похибка регресії втрат;

– mean_squared_error() – середня абсолютна похибка регресії втрат;

– mean_squared_log_error() – середньоквадратична логарифмічна похибка регресії втрат;

– median_absolute_error() – медіана абсолютної похибки регресії втрат;

Основними аргументами, що включають перераховані функції, є:

– y_true – Істинні цільові значення;

– y_pred – Розрахункові цільові значення.

9. Запуск написаного коду та оцінка результатів. В залежності від використовуваного програмного забезпечення, запуск програмного коду буде відбуватися по-різному. В моєму випадку, я використовувала Google Calobaratory, оскільки він є зручнішим для написання та запуску коду, а також має безкоштовну версію з досить широким інструментарієм, що повністю задовольняє потреби даного дослідження. Також, оскільки програмний код є досить великим та потребує часу, аби дочекатися проходження усіх ітерацій, Google Calobaratory дає можливість використати апаратний прискорювач (hardware accelerator) GPU, що дійсно зменшує час очікування. Для порівняння, якщо у MatLab обрати 1000 епох, то програма може зависнути, або може взагалі з неї викинути.

10. Внесення змін до налаштувань. Якщо результати не задовольняють, необхідно внести зміни до налаштувань моделі або її функцій (повтор пунктів 2-9).

11. Збереження моделі. За допомогою TensorFlow можна легко зберегти модель, а потім відтворити її за необхідності. Код виглядає наступним чином та є універсальним за умови використання Google Calobaratory або Jupyter Notebook:

```
!pip install -q pyyaml h5py  # Необхідно для збереження моделі у форматі HDF5
# Зберігаємо всю модель у HDF5 файл
model.save('my_model.h5')
# Відтворюємо в точності ту саму модель, враховуючи ваги та оптимізатор
new_model = keras.models.load_model('my_model.h5')
```

12. Використання моделі.

Також за допомогою бібліотеки sklearn можна використовувати вже готові моделі. Наприклад, лінійну регресію чи логістичну регресію. Перевагою такого методу є отримання також отримання рівняння регресії. Алгоритм використання досить простий:

– імпорт необхідних бібліотек;

– створення моделі;

– налаштування моделі;

– компіляція моделі;

– навчання моделі;

– оцінка моделі;

– отримання коефіцієнтів рівняння регресії;

– використання моделі.

Отже, TensorFlow широко використовується і для вирішення більш складних задач, ніж прогнозування параметрів. Дана платформа використовується для розпізнавання голосу, звуку, зображень, прогнозування часових рядів тощо. Платформа TensorFlow разом з бібліотекою Keras надає велику кількість бібліотек та модулів, а також документації, що дозволяє налаштовувати параметри більш усвідомлено. Для більш простого та послідовного створення штучних нейронних мереж розроблено алгоритм її програмування, а також розглянуто окремо кожний пункт.

Висновки до розділу

В даному розділі встановлено, що TensorFlow є досить зручною у використанні, а також досить гнучкою та багатофункціональною платформою. Вона успішно використовується для вирішення багатьох складних задач: розпізнавання голосу, для перетворення звуку у текст, розпізнавання зображень, для аналізу часових рядів, для виявлення руху тощо. Також є програмні продукти, що було створено за допомогою TensorFlow.

TensorFlow має великий асортимент програмних модулів, що й роблять її такою функціональною та гнучкою. Дані програмні модулі дозволяють досить точно створювати штучну нейронну мережу, аби вона якнайкраще описувала наявні дані.

В розділі створено алгоритм моделювання засобами TensorFlow для розв’язання задач регресії. Даний алгоритм досить простий та складається з 12 пунктів.

Отже, аналіз платформи TensorFlow довів, що вона є добрим інструментом для моделювання багатьох задач, в тому числі і задач регресії. Тому вирішено, що використання TensorFlow для вирішення задач моделювання процесу обробки матеріалів для прогнозування параметрів є доцільним.

РОЗДІЛ 4.
ПРАКТИЧНЕ ЗАСТОСУВАННЯ TENSORFLOW ДЛЯ МОДЕЛЮВАННЯ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ТА СИСТЕМ

Оброблення матеріалів – це базовий процес у приладобудуванні. Основними розрахунковими параметрами при обробці матеріалів є глибина різання, подача, швидкість різання, кількість обертів шпинделю верстата, його потужність тощо. Отже, задачею є моделювання процесу обробки різних матеріалів за допомогою платформи TensorFlow разом з бібліотекою Keras, а також порівняння з результатами моделювання тих самих параметрів за допомогою пакету NeuralNetwork Toolbox програми MatLab.

4.1. Моделювання періоду стійкості твердосплавних різців при обробленні титанових сплавів

Титанові сплави є важливими конструкційними матеріалами, які відзначаються великою міцністю, корозійною стійкістю і невеликою питомою вагою. Це визначає їх використання для виготовлення відповідальних деталей в приладо- та машинобудуванні. В даний час встановлено, що низька оброблюваність титанових сплавів пояснюється високою міцністю, низькою теплопровідністю, підвищеним нахилом до наклепу та великим питомим тиском, що пов’язаний з малою площею контакту стружки з різальним інструментом [28].

Тому важливою задачею технологічної підготовки виробництва (ТПВ) є дослідження процесу обробки титанових сплавів з точки зору визначення їх оброблюваності. Розв’язати дану задачу можливо шляхом виконання експериментальних досліджень з метою прогнозування вихідних параметрів процесу різання, що дозволять визначити оптимальні режими обробки. Основним параметром, що визначає оброблюваність титанового сплаву відповідним різальним інструментом є період стійкості різального інструменту. Для прогнозування періоду стійкості різального інструменту було використано дані дослідження періоду стійкості різців, оснащених пластинами із твердого сплаву ВК4, при токарній обробці титанового сплаву ВТ6. При проведенні досліджень швидкість різання V варіювалась в межах від 18 до 200 м/хв., подача S від 1 до 4 мм/хв., а глибина різання t – від 0,08 до 0,4 мм. При цьому період стійкості інструменту Т був в межах від 1,8 хв. до 27,1 хв. Фрагмент вхідних даних для навчання штучної нейронної мережі представлено в таблиці 4.1.

Таблиця 4.1. Фрагмент експериментальних даних для прогнозування періоду стійкості різців.

V, м/хв	70	100	140	90	160	120	100	150	120	160	90	28	18
t, мм	0,4	0,3	0,2	0,2	0,08	0,08	0,2	0,2	0,2	0,2	0,2	0,2	0,2
S, мм/хв	1	1	1	1	1	1	4	2	2	1	1	2	2
T, хв	7,5	5,4	4,5	16,2	5,8	14,9	7,8	2,3	7	3	27,1	3,1	16,3

Для навчання ШНМ застосовано пакет Neural Network Toolbox програми MatLab та алгоритм Levenberg-Marquardt. Для тренування (Training) використовувалося 70% даних, перевірка (Validation) та тестування (Test) – 15 % кожний. Кількість прихованих шарів в даній роботі складало 50.

Також в роботі представлено результати прогнозування періоду стійкості різального інструменту при інших, довільних початкових даних, що надано в таблиці 4.2.

Таблиця 4.2. Результати прогнозування періоду стійкості різального інструменту.

V, м/хв	18	160	140	160
t, мм	0,2	0,2	0,2	0,2
S, мм/хв	2	2	1	1
T, хв	16,3	20,24	5,62	3

За отриманими результатами дослідження періоду стійкості різального інструменту із твердого сплаву ВК4 при токарній обробці титанового сплаву ВТ6 можна зробити висновок, що використання штучних нейронних є ефективним засобом моделювання технологічних параметрів.

В [32] описано метод моделювання даного процесу за допомогою тих самих даних, проте із використанням TensorFlow. Для обробки експериментальних даних виконано їх попередню підготовку. Оскільки значення швидкості досить сильно відрізняються від значень інших параметрів, то для зручності та покращення навчання штучної нейронної мережі їх було зменшено в 100 разів. В даній роботі при моделюванні періоду стійкості твердосплавних різців при обробленні титанових сплавів було отримано кращі результати, ніж при моделюванні за допомогою програми MatLab. Фрагмент результатів прогнозування тренувальних та тестових даних надано в таблицях 4.3 та 4.4 відповідно.

Таблиця 4.3. Фрагмент результатів прогнозування тренувальних даних.

V, м/хв	S, мм/хв	t, мм	T (цільове), хв	T (прогн), хв
100	0.4	0.1	1.8	1.89
70	0.4	0.1	7.5	7.13
120	0.3	1.0	2.7	2.88
200	0.08	1.0	2.9	2.505
150	0.2	2.0	2.3	1.94
120	0.2	2.0	7.0	7.16

Таблиця 4.4. Фрагмент результатів прогнозування тестових даних.

V, м/хв	S, мм/хв	t, мм	T (цільове), хв	T (прогн), хв
90	0,4	0,1	4,2	3.64
100	0,3	1,0	5,4	6.37
100	0,2	2,0	11,0	10.66

На мою думку, зменшення похибки при моделюванні за допомогою TensorFlow пов’язано зі збільшенням кількості параметрів для налаштування мережі, що дало змогу підібрати більш доцільні в даному випадку. Також перевагою є те, що було отримано рівняння регресії, яке в подальшому можна використовувати без використання самої моделі, а також перевагою є можливість перевірки адекватності отриманого рівняння за допомогою критерія Фішера, що показав її адекватність з довірливою вірогідністю 0,95.

Програмний код моделювання періоду стійкості твердосплавних різців при обробленні титанових сплавів надано в Додатку А.

За отриманими результатами зроблено висновок, що для вирішення задачі моделювання періоду стійкості різального інструменту з твердого сплаву ВК4 при токарній обробці титанового сплаву ВТ6 доцільним є використання саме платформи TensorFlow. Дані результати також показали, що платформа TensorFlow є ефективним засобом для проектування, створення та навчання моделей в області машинного навчання.

4.2. Дослідження оброблюваності жаростійкої сталі фрезеруванням за різних режимів різання та вибір оптимального алгоритму навчання штучної нейронної мережі

Жаростійкі сталі та сплави – це високолеговані конструкційні матеріали, фізико-механічні характеристики яких є стабільними та лише незначно змінюються за високих температур. Дані матеріали мають високу стійкість проти хімічного руйнування в газовому середовищі, добре працюють в ненавантаженому та слабко навантаженому станах [31].

Основними особливостями процесу обробки жаростійких сталей та сплавів є [31]:

– високе зміцнення матеріалу в процесі деформації різанням;

– мала теплопровідність оброблюваного матеріалу, що призводить до підвищення температури в області контакту, та як наслідок, до дифузії та адгезії, інтенсивному зносу ріжучої кромки;

– здатність зберігати вихідну твердість та міцність за підвищених температур, що призводить до високих граничних навантажень в зоні контакту поверхні деталі та інструменту в процесі різання;

– знижена стійкість до вібрацій руху різання, що обумовлено високою зміцнюваністю жаростійких матеріалів за нерівномірного протікання процесу їх пластичної деформації.

Моделювання процесу торцевого фрезерування за результатами проведених експериментальних досліджень виконувалось з метою прогнозування потужності, що витрачається на фрезерування площинних поверхонь торцевими фрезами, що виготовлені з швидкорізальної сталі Р6М5К5, заготовок зі жаростійкої сталі 10Х11Н20Т3Р, при різних параметрах обробки в залежності від діаметру та ширини фрези, глибини та швидкості різання та подачі. При фрезеруванні глибина різання t змінювалась в межах від 1 до 3 мм, подача S – від 75 до 150 мм/хв., швидкість різання V від 19,0 до 41,5 мм/хв, а потужність приводу верстату змінювалась в межах від 0,40 до 4,01 кВт в залежності від режимів різання та діаметру фрези [29].

Фрагмент даних експериментальних досліджень визначення потужності, що витрачається на фрезерування площинних поверхонь торцевими фрезами різних діаметрів заготовок зі жаростійкої сталі 10Х11Н20Т3Р, надано в таблиці 4.5.

Для виконання поставленої задачі було згенеровано програмою MatLab скрипт для кожного з трьох методів навчання нейронної мережі: алгоритм зворотного розповсюдження Левенберга-Марквардта (Levenberg-Marquardt); алгоритм маштабуємих спряжених градієнтів (Scaled Conjugate Gradient); алгоритм байєсової регуляризації (Bayesian Regularization).

Для методу Levenberg-Marquardt було згенеровано програмою MatLab такий скрипт:

```
x = in';        %   input data
t = out';       %   output data
trainFcn = 'trainlm';       %   Levenberg-Marquardt backpropagation is usually fastest.
% Create a Fitting Network
```

```
hiddenLayerSize = 40;
net = fitnet(hiddenLayerSize,trainFcn);
% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,x,t);    % Train the Network
y = net(x);   % Test the Network
e = gsubtract(t,y);
performance = perform(net,t,y)
view(net)     % View the Network
```

Таблиця 4.5. Фрагмент даних експериментальних досліджень визначення потужності при фрезеруванні площинних поверхонь торцевими фрезами.

D фрези, мм	B фрези, мм	t, мм	S, мм/хв	V, мм/хв	N, кВт
80	55	1	80	41,5	0,4
80	55	2	120	30,5	1,19
80	55	3	65	33	0,97
125	85	1	95	33	0,73
125	85	2	85	29	1,3
125	85	3	75	26,5	1,72
160	100	2	125	23	2,48
160	100	2	150	21	2,97
200	120	1	120	25	1,3
200	120	2	105	22	2,1
200	120	2	128	20	2,56
200	120	3	96	20	2,85
200	120	3	110	19	3,1

Розкид похибок при застосуванні методу навчання за Levenberg-Marquardt зображено на рисунку 4.1.

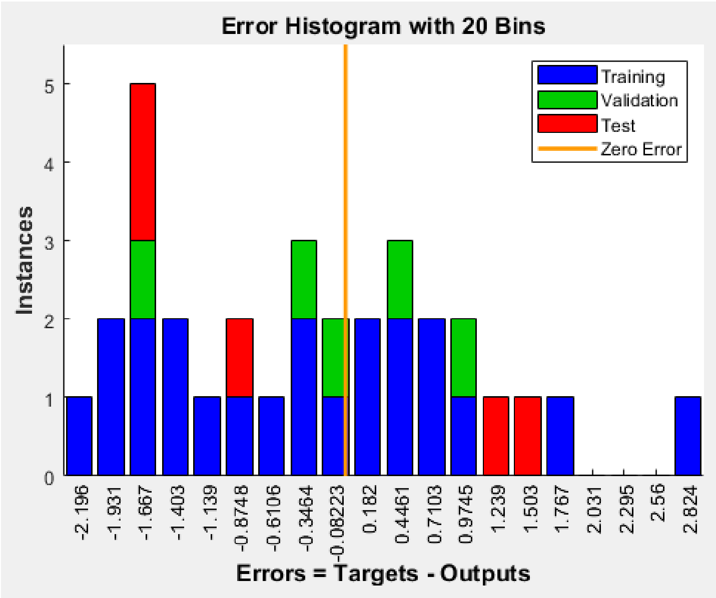


Рис. 4.1. Гістограма розкиду похибок навчання за алгоритмом Levenberg-Marquardt.

Для методу Bayesian Regularization було застосовано наступний скрипт:

```
x = in';
t = out';
trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation takes longer but may be better for challenging problems.
hiddenLayerSize = 40;
net = fitnet(hiddenLayerSize,trainFcn);
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,x,t);
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
```

view(net)

Розкид похибок за методом навчання Bayesian Regularization графічно ілюструє рисунок 4.2.

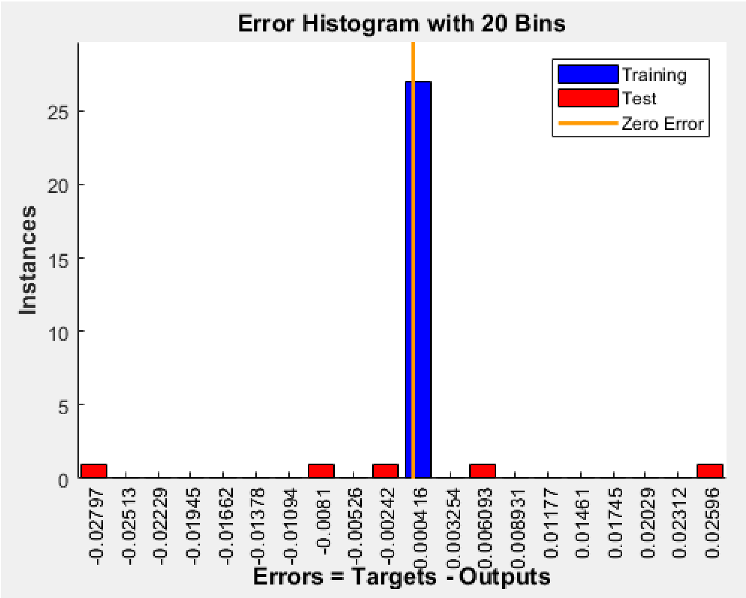


Рис. 4.2. Гістограма розкиду похибок навчання за алгоритмом Bayesian Regularization.

Для навчання за методом Scaled Conjugate Gradient було використано скрипт, що згенерований програмою MatLab:

x = in';   t = out';

trainFcn = 'trainscg'; % Scaled Conjugate Gradient backpropagation uses less memory. Suitable in low memory situations.

hiddenLayerSize = 40;

net = fitnet(hiddenLayerSize,trainFcn);

net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;

[net,tr] = train(net,x,t);

y = net(x);

e = gsubtract(t,y);

performance = perform(net,t,y)

view(net)

На рисунку 4.3 зображено розкид похибок навчання після застосування даного методу.

В таблиці 4.6 представлено результати прогнозування потужності процесу фрезерування за методом Bayesian Regularization, оскільки даний метод дав найменший інтервал розкиду похибок.

Отже, для навчання нейронної мережі при моделюванні процесу фрезерування жаростійкої сталі доцільно застосовувати алгоритм Bayesian Regularization.

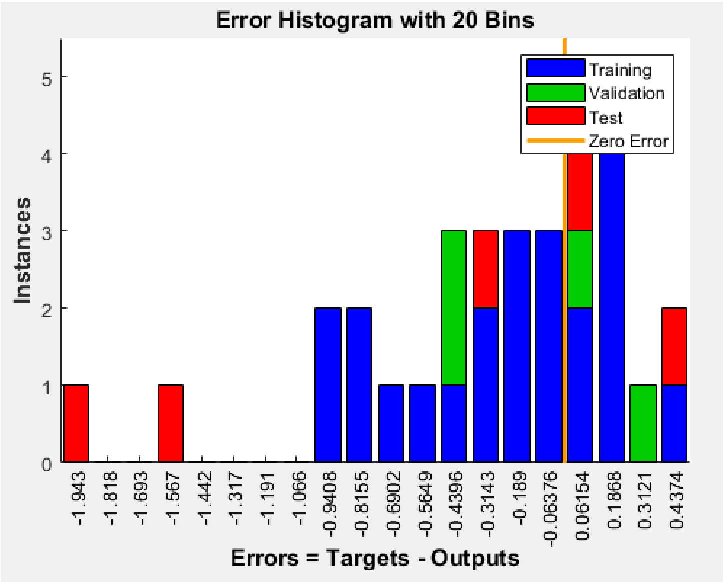


Рис. 4.3. Гістограма розкиду похибок навчання за алгоритмом Scaled Conjugate Gradient.

Таблиця 4.6. Результати прогнозування потужності процесу фрезерування.

D, мм	B, мм	t, мм	S, мм/хв.	V, мм/хв.	N (прогн), кВт
160	100	3	115	21	3,4200
80	55	2	165	27,5	1,6300
200	100	3	135	23,3	3,4910
150	60	2,3	75	21	2,1742
130	75	1,5	96	41,5	0,1957
160	80	3	150	40	1,5506

В даній роботі встановлено, що навчання штучних нейронних мереж при використанні різних алгоритмів на однакових даних дало різні значення та діапазон можливих похибок. Проведені дослідження дозволили створити штучну нейронну мережу, що моделює процес торцевого фрезерування жаростійкої сталі, і встановити, що для поставленої задачі доцільно застосовувати алгоритм Bayesian Regularization.

Також обробку результатів досліджень з метою моделювання процесу фрезерування та наступного прогнозування потужності різання виконувалось за допомогою платформи TensorFlow [31].

Створення моделі здійснювалося за алгоритмом, що було розроблено в даній роботі.

Оцінювання моделі виконувалось за допомогою формул середньоквадратичної похибки, середньої абсолютної похибки та відхилення від середнього (дисперсії). Проте в даних результатах я більше орієнтувалася на середньоквадратичне відхилення. В результаті обробки результатів досліджень за допомогою TensorFlow визначені похибки обчислень даних для тренування [31]:

MAE = 0.04457477154555144;

MSE = 0.003143211012646781 5;

RMSE = 0.05606434707233092 6.

та похибки обчислень тестових даних:

MAE = 0.07362550377845761;

MSE = 0.006938523085845853;

RMSE = 0.08329779760501387.

Фрагмент результатів експериментальних досліджень представлено в таблиці 4.7. При цьому блакитним кольором позначено дані, що були тестовими.

Програмний код моделювання процесу фрезерування заготівок з жаростійкої сталі надано у Додатку Б.

Вирішення задачі моделювання процесу фрезерування плоских поверхонь заготовок з жаростійкої сталі 10Х11Н20Т3Р торцевими фрезами різних діаметрів шляхом виміру потужності, що витрачається для обробки різанням, показало, що створення моделі за допомогою платформи TensorFlow на основі формування ШНМ, є ефективним методом вирішення подібних задач моделювання технологічних процесів [31].

Таблиця 4.7. Фрагмент результатів експериментальних досліджень прогнозування потужності різання.

D, мм	B, мм	t, мм	S, мм/хв	V, мм/хв	N (ціл), кВт	N (прог), кВт
80	55	1	190	32	0,94	0.94
80	55	2	120	30,5	1,19	1.13
80	55	2	165	27,5	1,63	1.6
80	55	3	65	33	0,97	0.94
125	85	1	95	33	0,73	0.7
125	85	1	125	29,5	0,96	0.96
125	85	2	85	29	1,3	1.35
125	85	2	110	26	1,68	1.75
125	85	2	150	23,5	2,3	2.29
125	85	3	100	24	2,3	2.3
125	85	3	140	21,5	3,22	3.19
160	100	2	125	23	2,48	2.34
160	100	3	135	19	4,01	3.95
200	120	1	120	25	1,3	1.26
200	120	1	145	23	1,6	1.57
200	120	2	105	22	2,1	2.09
200	120	2	120	21	2,38	2.35
200	120	3	110	19	3,1	3.15

4.3. Вибір оптимальної штучної нейронної мережі за структурою та методом навчання

Для вирішення задачі вибору оптимальної структури штучної нейронної мережі за кількістю прихованих шарів штучної нейронної мережі та ефективного методу її навчання для розв’язання конкретної технологічної задачі використано результати процесу фрезерування площинних поверхонь заготівок із корозійностійкої сталі 40Х13 торцевими фрезами з пластинами із твердого сплаву ВК6М без охолодження. При проведенні експериментальних досліджень діаметр фрези в даному дослідженні варіювався в межах від 100 до 200 мм, ширина фрези – від 70 до 120 мм, глибина різання t – від 1 до 5 мм, швидкість різання – від 66 до 164 мм/хв, подача – від 140 до 800 мм/хв, а потужність – від 1,13 до 8,5 кВт [30]. Фрагмент вхідних даних за результатами експериментальних досліджень визначення потужності процесу фрезерування площинних поверхонь заготівок із корозійностійкої сталі 40Х13 торцевими фрезами з пластинами із твердого сплаву ВК6М без охолодження представлено в таблиці 4.8.

Таблиця 4.8. Фрагмент результатів досліджень процесу фрезерування площинних поверхонь торцевими фрезами заготовок із сталі 40Х13.

D фрези, мм	B фрези, мм	t, мм	V, мм/хв	S, мм/хв	N, кВт
100	70	1	164	210	1,13
100	70	2	132	170	2,25
100	70	5	93	236	6,33
125	85	1	124	800	1,31
125	85	5	78	190	6,22
160	110	1	121	190	1,61
160	110	5	88	140	5,93

Скрипти, що створює програма MatLab, для різних методів навчання, що розглядаються в даній роботі аналогічні скриптам, що представлені в попередніх дослідженнях.

На рисунках 4.4-4.12 представлено гістограми розподілу похибок при тренуванні, валідації та тестуванні штучних нейронних мереж методами Levenberg-Marquardt, Bayesian Regularization, Scaled Conjugate Gradient Кількість прихованих шарів в кожному методі варіювалась: 10, 50, 100. Це дало можливість дослідити, як впливає кількість прихованих шарів на точність прогнозування параметрів.

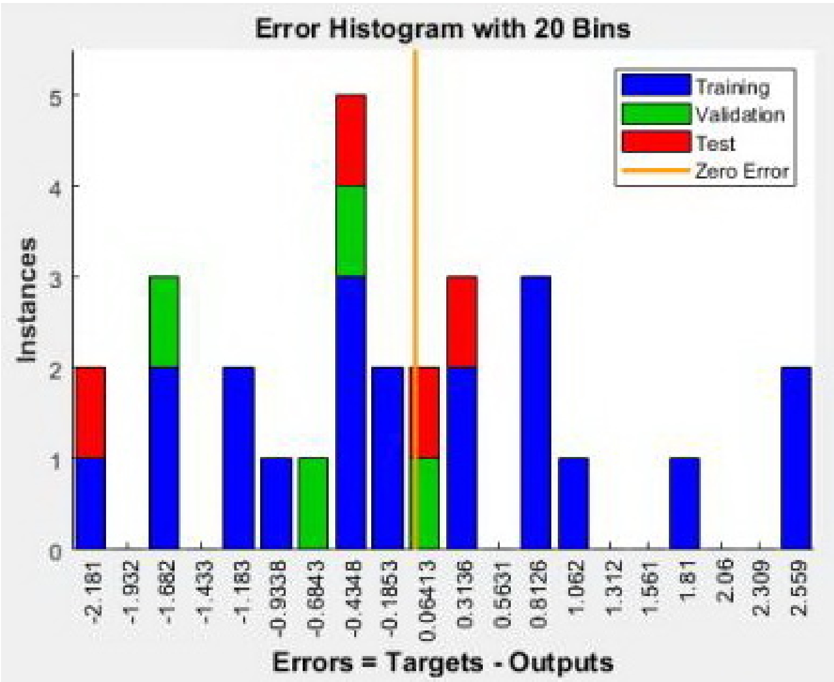


Рис. 4.4. Гістограма розкиду похибок за методом навчання Levenberg-Marquardt при застосуванні 10 прихованих шарів.

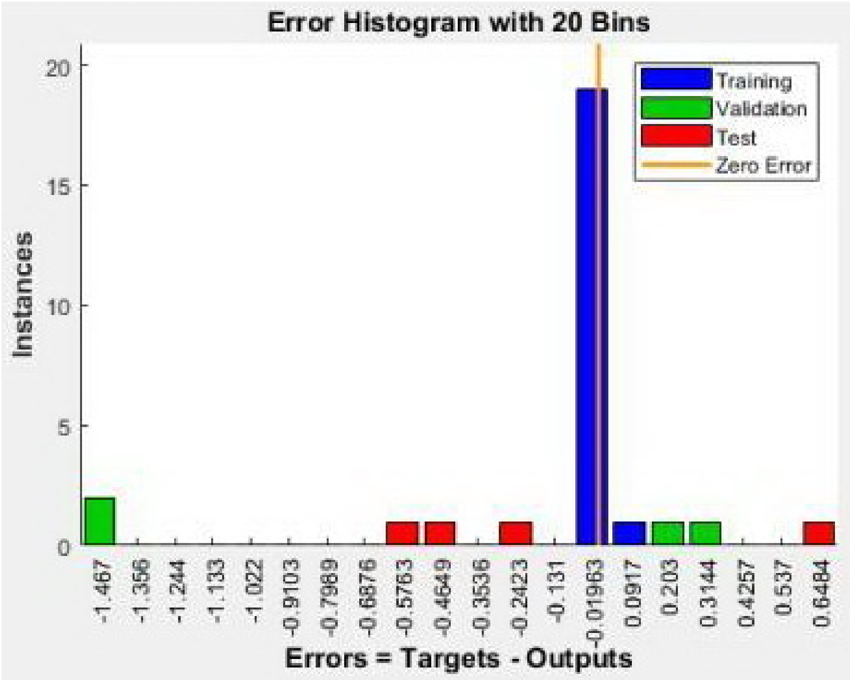


Рис. 4.5. Гістограма розкиду похибок за методом навчання Levenberg-Marquardt при застосуванні 50 прихованих шарів.

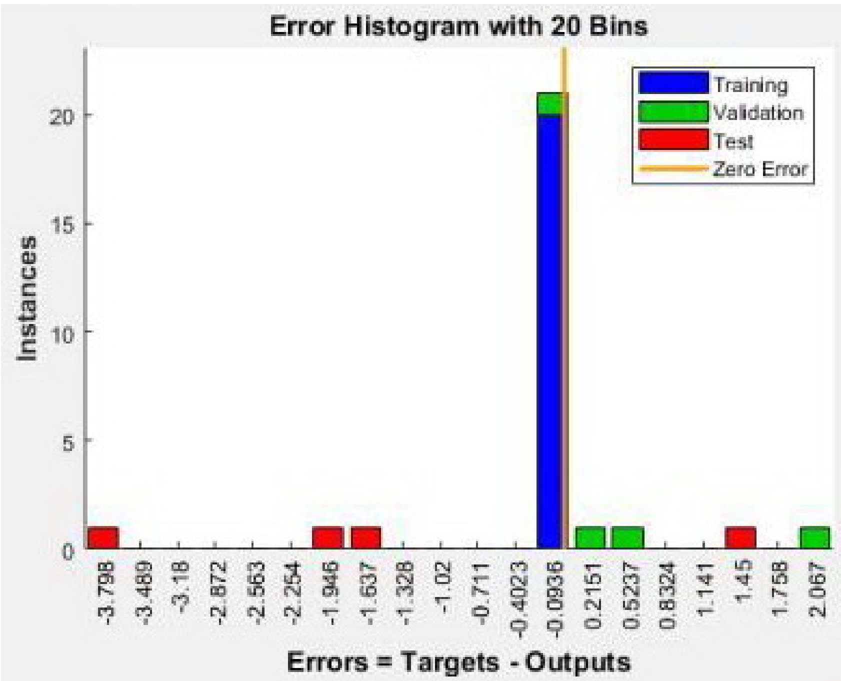


Рис. 4.6. Гістограма розкиду похибок за методом навчання Levenberg-Marquardt при застосуванні 100 прихованих шарів.

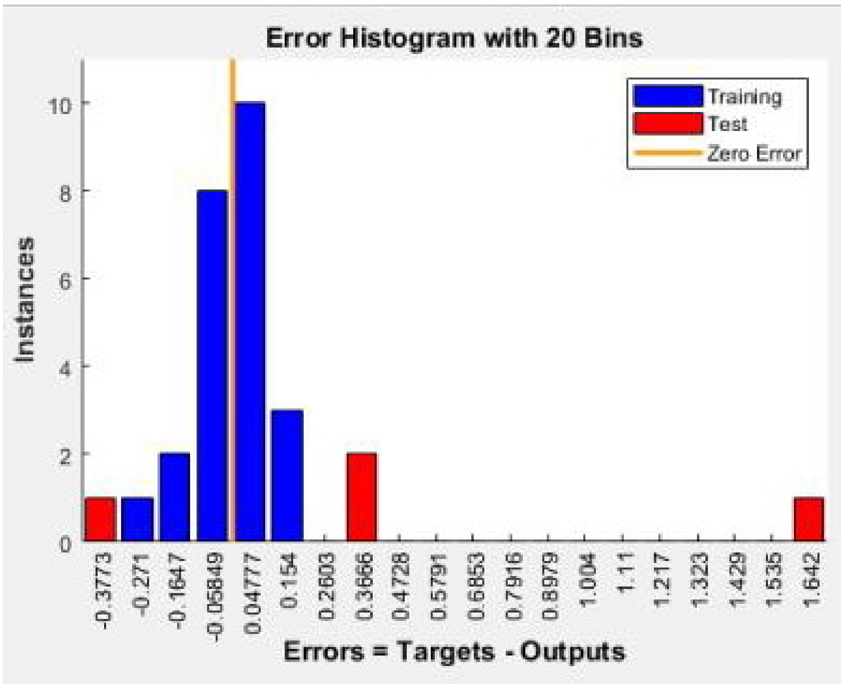


Рис. 4.7. Гістограма розкиду похибок за методом навчання Bayesian Regularization при застосуванні 10 прихованих шарів.

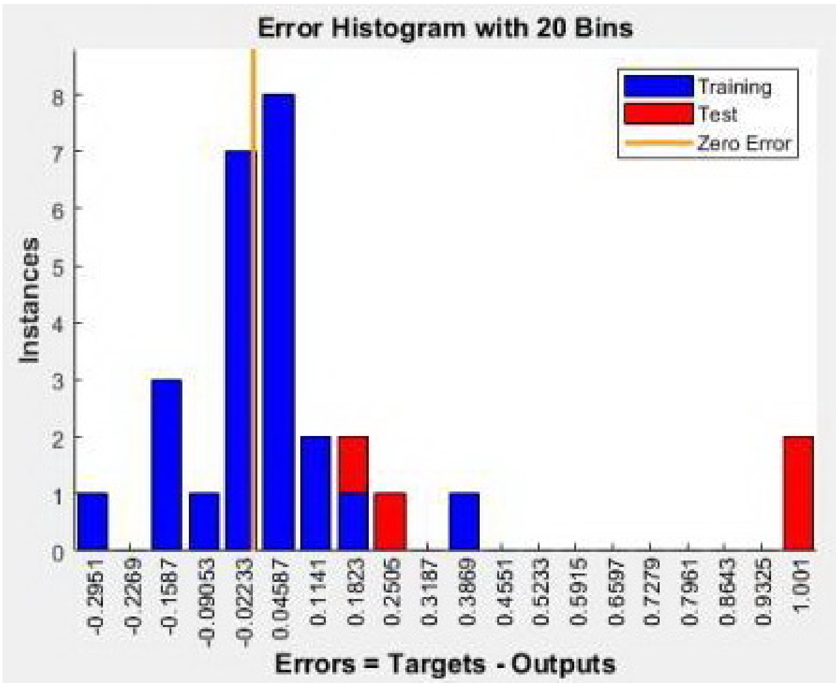


Рис. 4.8. Гістограма розкиду похибок за методом навчання Bayesian Regularization при застосуванні 50 прихованих шарів.

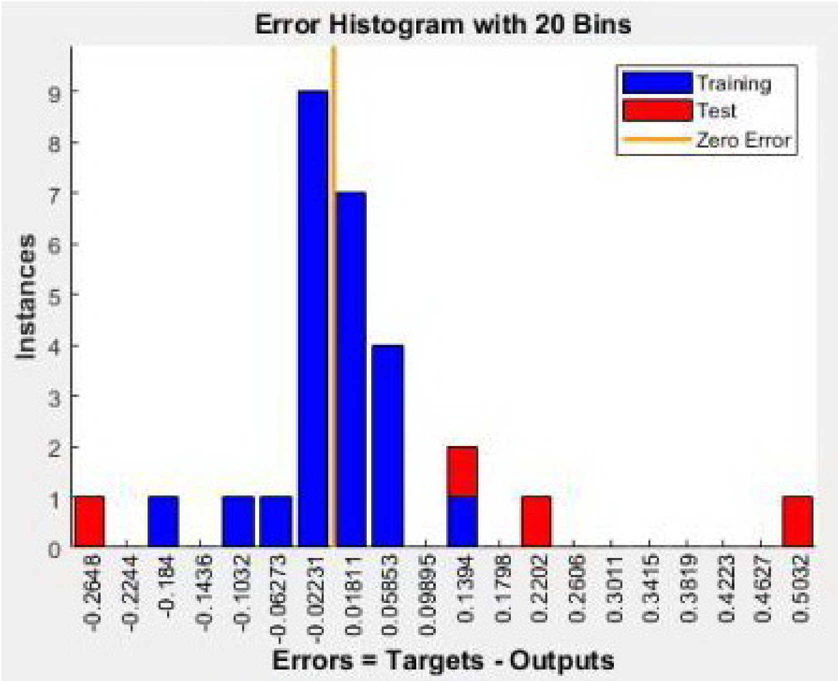


Рис. 4.9. Гістограма розкиду похибок за методом навчання Bayesian Regularization при застосуванні 100 прихованих шарів.

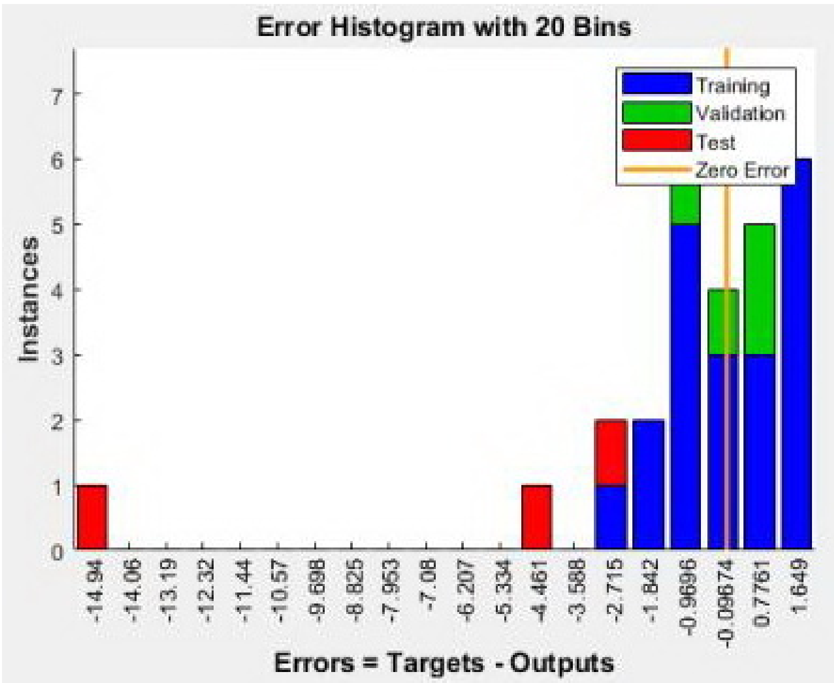


Рис. 4.10. Гістограма розкиду похибок за методом навчання Scaled Conjugate Gradient при застосуванні 10 прихованих шарів.

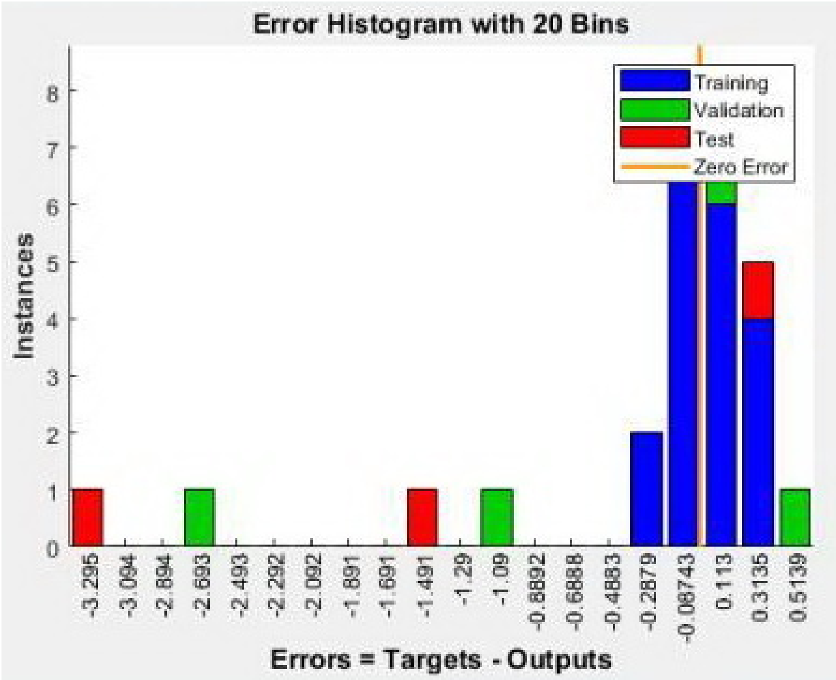


Рис. 4.11. Гістограма розкиду похибок за методом навчання Scaled Conjugate Gradient при застосуванні 50 прихованих шарів.

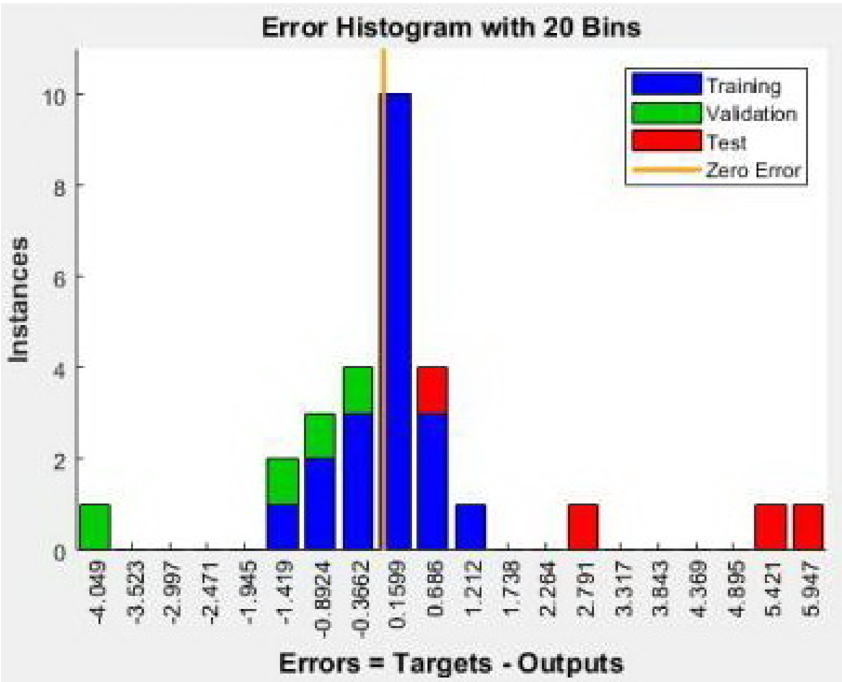


Рис. 4.12. Гістограма розкиду похибок за методом навчання Scaled Conjugate Gradient при застосуванні 100 прихованих шарів.

В таблиці 4.9 представлено діапазони значень похибок навчання штучної нейронної мережі трьома методами: Levenberg-Marquardt, Bayesian Regularization, Scaled Conjugate Gradient, за різної кількості прихованих шарів – 10, 50 та 100. Подальше зменшення або збільшення кількості шарів є невиправданим, оскільки через малу кількість прихованих шарів штучна нейронна мережа не встигне навчитися, а велика кількість прихованих шарів може призвести до невиправданого збільшення кількості часу на навчання, а також до перенавчання мережі, що проявляється в тому, що мережа буде гарно працювати на навчальній вибірці, але дуже погано на вхідних прикладах, що не входять до неї [75].

Аналіз похибок навчання ШНМ різними методами та за різною кількістю прихованих шарів (рис. 4.4-4.12), показав, що найкращим методом, який дає найменший діапазон розкиду похибки, для вирішення задачі моделювання процесу фрезерування площинних поверхонь заготівок із корозійностійкої сталі 40Х13 торцевими фрезами з пластинами із твердого сплаву ВК6М без охолодження при будь-якій кількості прихованих шарів є метод Bayesian Regularization, а метод Scaled Conjugate Gradient – дає найбільший діапазон розкиду похибки [30].

Таблиця 4.9. Діапазони значень похибок навчання ШНМ різними методами та за різною кількістю прихованих шарів.

Кількість прихованих шарів	Метод навчання мережі		
	Levenberg-Marquardt	Bayesian Regularization	Scaled Conjugate Gradient
10	[-2.181; 2.559]	[-0.3773; 1.642]	[-14.94; 1.649]
50	[-1.467; 0.6484]	[-0.2951; 1.001]	[-3.295; 0.5139]
100	[-3.798; 2.067]	[-0.2648; 0.5032]	[-4.049; 5.947]

Проте для кожного розв’язуваної задачі необхідно перевіряти найбільш прийнятну кількість прихованих шарів, що забезпечить найменшу похибку прогнозування, оскільки, як було представлено у попередніх прикладах, похибка може бути значно меншою і при 40 прихованих шарах. Результати роботи штучної нейронної мережі за методом Bayesian Regularization при 100 прихованих шарах представлено в таблиці 4.10.

Таблиця 4.10. Результати роботи штучної нейронної мережі за методом Bayesian Regularization при 100 прихованих шарах.

D фрези, мм	B фрези, мм	t, мм	V, мм/хв	S, мм/хв	N (очік), кВт	N (прогн), кВт
100	70	1	164	210	1.13	1.1073
100	70	2	90	340	5.5	5.4907
125	85	5	90	145	4.85	4.8405
160	110	1	121	190	1.61	1.6512
200	120	1	84	162	4.5	4.3793
200	120	2	66	190	8.5	8.4267

Проте для забезпечення меншої похибки було також змодельовано даний процес за допомогою платформи TensorFlow та бібліотеку Keras. І даній роботі вхідні дані представлялися по-різному: спочатку незмінні, потім – стандартизовані, та наостанок – нормалізовані. Вихідні дані завжди стандартизувалися. Також в даній роботі створено користувацькі функції get_dataset(), що відповідає за підготовку даних; create_model(), що відповідає за створення моделі та ї налаштування та компіляцію; evaluate_model(), що тренує та оцінює модель; repeated_evaluation() відповідає за повтор функції evaluate_model() та оцінки остаточної моделі; calc_error() дозволяє швидко оцінити результати прогнозування моделі за допомогою MAE, MSE, RMSE.

Для отримання кращих результатів модель навчалася 30 разів по 1200 епох. В даному випадку використовувалася функція зворотного виклику ModelCheckpoint, що зберігає у файл тільки найкращі ваги, та їх можна завантажити вкінці навчання для оцінки результатів. Результати прогнозування занесені до таблиці 4.11, де блакитним виділено дані для тестування. Візуалізацію зведених даних по структурі ШНМ можна бачити на рисунку 4.13.

В таблиці 4.12 представлено похибки прогнозування даних для тренування (білий колір) та даних для тестування (блакитний). Програмний код надано у Додатку В.

Аналіз результатів моделювання показав, що найкращим є застосування стандартизації до вхідних та вихідних даних.

Таблиця 4.11. Результати прогнозування потужності фрезерування за різних методів підготовки даних.

D, мм	B, мм	t, мм	V, об/хв	S, мм/об	N, кВт	Unscaled	Normalized	Scaled
100	70	1	164	210	1,13	1,126	0,985	0,992
100	70	1	112	425	2,29	2,320	2,364	2,282
100	70	2	132	170	2,25	2,257	2,427	2,471
100	70	2	103	265	4,28	3,951	3,830	3,627
100	70	2	90	340	5,5	5,450	5,090	5,309
125	85	1	124	800	1,31	1,312	1,374	1,347
125	85	2	99	155	2,94	2,797	2,951	2,915
125	85	2	86	210	4,12	3,828	4,149	4,171
125	85	2	80	245	4,71	4,657	4,822	4,828
125	85	5	78	190	6,22	6,234	6,397	6,373
125	85	5	72	220	7,2	7,189	7,058	7,175
160	110	2	84	200	5,08	5,123	5,102	5,097
160	110	2	73	260	6,61	7,049	6,449	6,733
160	110	5	77	180	7,62	7,611	7,377	7,434
200	120	1	73	210	5,8	5,828	5,994	6,033
200	120	1	66	265	7,5	5,314	6,892	7,126
200	120	2	76	145	6,5	6,482	6,640	6,590
200	120	2	66	190	8,5	8,451	7,821	8,024

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_37 (Dense)             multiple                  180
_________________________________________________________________
dense_38 (Dense)             multiple                  1550
_________________________________________________________________
dense_39 (Dense)             multiple                  2550
_________________________________________________________________
dense_40 (Dense)             multiple                  51
=================================================================
Total params: 4,331
Trainable params: 4,331
Non-trainable params: 0
```

Рис. 4.13. Результат виконання функції summary().

Таблиця 4.12. Похибки прогнозування параметрів.

Похибка	Unscaled	Normalized	Scaled
MAE	0,01864	0,07818	0,05538
MSE	0,00067	0,01075	0,00535
RMSE	0,02596	0,10366	0,07317
MAE	0,81137	0,31195	0,30026
MSE	1,29054	0,14959	0,14570
RMSE	1,13602	0,38677	0,38171

4.4. Рекомендації з використання результатів досліджень

Для моделювання процесів та систем доцільно використовувати штучні нейронні мережі. Вони допомагають вирішувати цілий спектр різноманітних задач, серед яких, зокрема, задачі прогнозування, регресії, апроксимації, керування тощо.

Проте яку саме архітектуру необхідно обрати – залежить від поставленої задачі та типу навчання. Багатошарові мережі штучні нейронні мережі, наприклад, найчастіше використовуються для задач апроксимації, класифікації та прогнозування, рідше – багатошарові штучні нейронні мережі зі зворотнім зв’язком чи одношарові. Тип навчання даних ШНМ – “з вчителем”.

Для розв’язання задач моделювання процесів та систем також можна використовувати штучні нейронні мережі з типом навчання “без вчителя”. Зазвичай це рекурентні. Вони можуть використовуватися для вирішення комбінаторних задач оптимізації та асоціативної пам’яті.

Перед тим, як моделювати штучну нейронну мережу для вирішення певної задачі, необхідно визначити, яка саме архітектура та тип навчання будуть доцільні для її вирішення.

Отже, в таблиці 4.13 наведено рекомендації щодо застосування штучних нейронних мереж в залежності від типу навчання та поставленої задачі.

Проте для моделювання штучних нейронних мереж також необхідно обрати інструментарій, який має певні системні вимоги. Для використання засобів TensorFlow при розробленні ШНМ необхідно, аби були виконані наступні вимоги [77]:

1. Вимоги до обладнання:

– Графічна карта NVIDIA GPU з підтримкою CUDA та обчислювальною потужністю 3.5 та вище.

2. Вимоги до програмного забезпечення:

– драйвери NVIDIA GPU – для CUDA 10.1 необхідно 418.x та вище;

– інструментарій CUDA – TensorFlow підтримує CUDA 10.1;

– CUPTI надається разом з інструментарієм CUDA;

3. Системні вимоги:

– Python 3.5 – 3.8;

– pip 19.0 або пізніша версія;

– Ubuntu 16.04 або пізніша версія (64-bit);

– macOS 10.12.6 (Sierra) або пізніша версія (64-bit) (немає підтримки GPU);

– Windows 7 або пізніша версія (64-bit);

– Raspbian 9.0 або пізніша версія;

– для підтримки GPU необхідна карта з підтримкою CUDA (Ubuntu та Windows).

Таблиця 4.13. Рекомендації щодо застосування ШНМ [76].

Тип навчання	Архітектура	Задачі
З вчителем	Одно- та багатошарові мережі з прямим та зворотнім зв’язком	Класифікація. Апроксимація функцій. Асоціативна пам’ять. Прогнозування. Керування.
	Багатошарові мережі з прямим зв’язком	Аналіз даних. Класифікація.
	Мережі ART (мережі на базі теорії адаптивного резонансу)	Класифікація.
Без вчителя	Багатошарові мережі з прямим зв’язком	Кластеризація. Аналіз даних.
	Мережі Хопфілда	Асоціативна пам’ять. Оптимізація.
	Мережі SOF (самоорганізуючі мапи, мережі Кохонена)	Кластеризація. Аналіз даних.
	Мережі ART (мережі на базі теорії адаптивного резонансу)	Кластеризація.
Мішана	Мережі RBF (мережі з радіальними базисними функціями) – часковий випадок двошарової мережі з прямим зв’язком	Класифікація. Апроксимація функцій. Прогнозування. Керування.

Проте є платформа від компанії Google, що називається Google Colaboratory. Вона також використовується в даній дипломній роботі. Платформа Google Colaboratory повністю заснована на хмарі та не потребує встановлення на комп'ютер або будь-яких внутрішніх вимог до програмного забезпечення чи потужності комп'ютеру. Все чого дана платформа потребує:

– стабільного Інтернет-підключення;

– браузер (Chrome, Mozilla, Opera тощо).

Дана платформа повністю підтримує TensorFlow, тому його заздалегідь встановлено та оптимізовано під використовуване обладнання. Також Google Colaboratory дозволяє змінювати тип середовища виконання на графічний процесор, що рекомендовано зробити, оскільки тоді код програми буде виконуватися швидше.

Висновки до розділу

За отриманими результатами дослідження періоду стійкості різального інструменту із твердого сплаву ВК4 при токарній обробці титанового сплаву ВТ6 показало, що застосування штучних нейронних мереж є ефективним засобом моделювання технологічних процесів, а для створення, навчання та вивчення моделей найкращим інструментом є платформа TensorFlow.

На основі дослідження потужності процесу фрезерування площинних поверхонь торцевими фрезами різних діаметрів заготовок з жаростійкої сталі 10Х11Н20Т3Р доведено, що створення моделі за допомогою TensorFlow на основі формування штучних нейронних мереж, є ефективним методом вирішення схожих задач моделювання ТП.

Дослідження потужності фрезерування площинних поверхонь заготовок із корозійностійкої сталі 40Х13 торцевими фрезами з пластинами із твердого сплаву ВК6М без охолодження показало, що найкращі результати можна отримати застосовуючи стандартизацію до вхідних та вихідних даних.

Отже, використання штучних нейронних мереж є ефективним засобом моделювання технологічних параметрів обробки різних матеріалів. В даному розділі також порівнюється моделювання за допомогою TensorFlow, а також за допомогою MatLab, а саме: Neural Network Toolbox. Зроблено висновок, що для отримання швидких результатів краще застосовувати MatLab, проте для отримання більш точної моделі, яку можна надалі успішно використовувати на виробництві, краще моделювати дані процеси із застосуванням TensorFlow.

ЗАГАЛЬНІ ВИСНОВКИ ДО ДИПЛОМНОЇ РОБОТИ

В дипломній роботі проаналізовано стан проектування та дослідження процесів і систем, що дозволило визначити основний метод їх реалізації – моделювання.

Розглянуто та проаналізовано різноманітні методи моделювання та їх класифікацію. Встановлено, що найкращим методом моделювання процесів і систем, до яких відноситься і технологічний процес, є математичне моделювання.

Огляд методів математичного моделювання встановив, що найбільш ефективним та перспективним методом математичного моделювання процесів і систем є використання ШНМ.

Аналіз задач, що можуть вирішені за допомогою ШНМ, та областей їх використання показав, що вони знайшли своє широке використання у багатьох сферах життя: економіка і бізнес, медицина, авіоніка, зв'язок тощо. Проте найбільш широкого використання ШНМ знайшли саме у промисловому виробництві. Вони використовуються для підвищення продуктивності, покращення логістики, зменшення кількості браку, скорочення кількості відмов обладнання, у автоматизованих лініях тощо. Існує велика кількість прикладів успішного використання ШНМ на виробництві, зокрема і у приладобудуванні. Тому для моделювання ТП та прогнозування необхідних параметрів обрано саме ШНМ. Також виконано аналіз програмних засобів та інструментарію для моделювання за допомогою ШНМ та сформульовано вимоги до програмних засобів, що дозволило обрати з них найбільш раціональний та зручний – платформу TensorFlow.

Розглянуто платформу TensorFlow, області її використання та задачі, що можна вирішити з її допомогою. Встановлено, що TensorFlow є багатофункціональною, гнучкою та зручною платформою для моделювання ШНМ. Також зроблено огляд доступних програмних модулів, що дозволяють точно і просто створювати власну ШНМ. Велика кількість документації та прикладів від компанії-розробника дає можливість самостійно створювати власні налаштування або розширювати існуючі для найбільш точного опису модельованого процесу. На основі аналізу усіх можливостей TensorFlow створено алгоритм моделювання процесів для прогнозування параметрів за допомогою TensorFlow, що стало підґрунтям для практичного використання TensorFlow при вирішенні задачі моделювання параметрів при ТПВ.

Використання показало, що TensorFlow є більш ефективним засобом для моделювання процесу обробки заготовки, ніж програма MatLab. ШНМ створені за допомогою TensorFlow дали більш точні результати, оскільки кількість змінюваних параметрів є значно більшою. Також в порівнянні з MatLab TensorFlow є більш зручним, легким та інтуїтивним, що надає йому значної переваги. Основним недоліком TensorFlow є час, що витрачається на налаштування ШНМ, оскільки необхідно найбільш вдало підібрати комбінації усіх параметрів для найбільш точного моделювання процесу.

Результати розв’язання задач, створені за допомогою TensorFlow, можуть бути успішно впроваджені при проектуванні технологічних процесів, що дозволить зробити ТПВ більш якісною.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Поняття моделі та моделювання. Властивості та класифікація моделей. НПУ імені М.П. Драгоманова : веб-сайт. URL: https://www.wiki.npu.edu.ua/index.php/ Поняття моделі та моделювання. Властивості та класифікація моделей.

2. Управління земельними ресурсами / за ред. проф. А.М. Третяка. Вінниця : Нова книга, 2006. 360 с. URL: https://buklib.net/books/35314/ .

3. 5.3 Моделирование производственных процессов и систем. Основы научной деятельности (Лежнева Е.И.): веб-сайт. URL: http://dl.khadi.kharkov.ua/mod/book/view.php?id=16277&chapterid=2916.

4. Белов, П. Г. Управление рисками, системный анализ и моделирование в 3 ч. Часть 1 : учебник и практикум для вузов. Москва : Издательство Юрайт, 2020. 211 с. URL: https://studme.org/1208070224511/menedzhment/upravlenie riskami sistemnyy analiz i modelirovanie

5. Соколова О.А., Вислоух С.П. Застосування методу евристичної самоорганізації моделей при розв'язанні технологічних задач. Погляд у майбутнє приладобудування : зб. праць XII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених. Київ : Центр учбової літератури, 2019. С. 190-192.

6. Глава 1. Цели и задачи математического моделирования процессов и систем. СтудИзба : веб-сайт. URL: https://studizba.com/lectures/129-inzhenerija/1910-matematicheskoe-modelirovanie-processov/37301-3-celi-i-zadachi-matematicheskogo-modelirovanija-processov-i-sistem.html

7. Горелик В.А. Теория принятия решений : учебное пособие для магистрантов. Москва : МПГУ, 2016. 152 с. URL: https://books.google.com.ua/books?id=4MxSDwAAQBAJ&pg=PA24&lpg=PA24&dq="схема"+применения+математических+методов+для+решения+реальных+задач&source=bl&ots=xCvcXSblDR&sig=ACfU3U375HZzMmueD_XITL-xXzPdH7AOFw&hl=ru&sa=X&ved=2ahUKEwjn766mq8rpAhVMlYsKHd1HCiIQ6AEwAnoECAkQAQ#v=onepage&q="схема" применения математических методов для решения реальных задач&f=false

8. Етапи математичного моделювання. StudFiles : веб-сайт. URL: https://studfile.net/preview/6012686/page:9/

9. Васильев К.К., Служивый М.Н. Математическое моделирование систем связи : учебное пособие. Ульяновск : УлГТУ, 2008. 170 с. URL: http://scask.ru/a_book_mm.php

10. 1 Основы теории погрешностей. Авдюшев В.А. Основы численных методов : веб-сайт. URL: http://www.astro.tsu.ru/OsChMet/1_1.html

11. Виды моделирования. Компьютерное моделирование. Л.Бахвалов : веб-сайт. URL: http://bourabai.kz/cm/bahvalov2.htm

12. Математичне програмування. Wiki ТНТУ : веб-сайт. URL: https://wiki.tntu.edu.ua/Математичне_програмування

13. Понятский В.М., Велешки С.И., Жирнова А.В. Использование метода группового учета аргументов для выбора структуры модели динамического объекта. Известия ТулГУ. Технические науки. 2013. №2. URL: https://cyberleninka.ru/article/n/ispolzovanie-metoda-gruppovogo-ucheta-argumentov-dlya-vybora-struktury-modeli-dinamicheskogo-obekta

14. Радченко С.Г. Математическое моделирование технологических процессов в машиностроении : монография. Київ : ЗАО “Укрспецмонтажпроект”, 1998. 274 с.

15. Душинський В.В., Кравченко С.Г. Моделювання та оптимізація в машинобудуванні : навч. посіб. Київ : НМК ВО, 1993. 296 с.

16. Ивахненко А.Г., Юрачковский Ю.П. Моделирование сложных систем по экспериментальным данным : монография. Москва : Сов радио, 1986. 120 с.

17. Равская Н.С. Разработка прогрессивных режущих инструментов на основе моделирования их работы методом самоорганизации : автореф. дис. на соискание уч. степени д-ра тех. наук : OD 71 92-5/121. Киев, 1991. 32 с.

18. Пономарев В.Б., Лошкарев А.Б. Математическое моделирование технологических процессов : курс лекций. Екатеринбург : ГОУ ВПО УГТУ–УПИ, 2006. 129 с. URL: https://study.urfu.ru/Aid/Publication/2525/1/Ponomarev_loshkarev.pdf

19. Математическое моделирование - вещь, которая способна решить многие проблемы в нашей жизни. HackMD : веб-сайт. URL: https://hackmd.io/s/r1MXZ_O3

20. Преимущества нейронных сетей. AIportal : веб-сайт. URL: http://www.aiportal.ru/articles/neural-networks/advantages.html

21. Белоглазов Д.А. Особенности нейросетевых решений, достоинства и недостатки, перспективы применения. Известия ЮФУ. Технические науки. 2008. № 7. URL: https://cyberleninka.ru/article/n/osobennosti-neyrosetevyh-resheniy-dostoinstva-i-nedostatki-perspektivy-primeneniya

22. Нейронная сеть. ВикипедиЯ : веб-сайт. URL: https://ru.wikipedia.org/wiki/Нейронная_сеть

23. What is an artificial neural network? Here's everything you need to know. Digital Trends : веб-сайт. URL: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/

24. Искусственные нейронные сети. Начальные сведения. EXPonenta : веб-сайт. URL: http://old.exponenta.ru/soft/others/mvs/stud3/3.asp

25. Нейронные сети. StatSoft : веб-сайт. URL: http://statsoft.ru/home/textbook/modules/stneunet.html

26. Introduction to Artificial Neural Networks(ANN). towards data science : веб-сайт. URL: https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9

27. Богославский С.Н. Область применения искусственных нейронных сетей и перспективы их развития. Научный журнал КубГАУ. 2007. №27. URL: https://cyberleninka.ru/article/n/oblast-primeneniya-iskusstvennyh-neyronnyh-setey-i-perspektivy-ih-razvitiya

28. Соколова О.А., Вислоух С.П. Моделювання періоду стійкості твердосплавних різців при обробленні титанових сплавів. Процеси механічної обробки, верстати та інструмент : зб. наук. праць X Всеукр. наук.-тех. конф. з міжнародною участю. Житомир : Державний університет «Житомирська політехніка». 2019. – С. 181-182.

29. Соколова О.А., Вислоух С.П. Моделювання процесу фрезерування деталей із жаростійкої сталі засобами штучних нейронних мереж. Ефективність інженерних рішень в приладобудуванні : зб. наук. праць XV Всеукр. наук.-практ.

конф. студентів, аспірантів та молодих вчених. Київ : Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2019. – С. 172-175.

30. Соколова О.А., Вислоух С.П. Вибір ефективних методів навчання штучних нейронних мереж при прогнозуванні технологічних параметрів. Modern questions of production and repair in industry and in transport : materials of the 20th International Scientific and Technical Seminar. Kosice, Tbilisi, Georgia, 2020. С. 180-183.

31. O. Sokolova, S. Vysloukh. Modeling of parameters of the milling of parts from heat-resistant steel. Погляд у майбутнє приладобудування : зб. праць XIII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених. Київ : Центр учбової літератури, 2020. С. 88-91.

32. O. Sokolova, S. Vysloukh. Modeling of titanium alloys processing parameters. Погляд у майбутнє приладобудування : зб. праць XIII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих вчених. Київ : Центр учбової літератури, 2020. С. 84-87.

33. Области практического применения искусственных нейронных сетей. NeuroPro : веб-сайт. URL: http://www.neuropro.ru/neu7.shtml

34. Иванько А.Ф., Иванько М.А., Сизова Ю.А. Нейронные сети: общие технологические характеристики. Научное обозрение. Технические науки. 2019. № 2. URL: https://science-engineering.ru/ru/article/view?id=1236

35. 5 Ways Machine Learning Is Leading to Smarter Manufacturing. business.com : веб-сайт. URL: https://www.business.com/articles/machine-learning-and-manufacturing/

36. Asset Monitoring & Predictive Maintenance. Deloitte Digital : веб-сайт. URL: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/about-deloitte/us-a-turnkey-iot-solution-for-manufacturing.pdf

37. Smartening up with Artificial Intelligence (AI) - What's in it for Germany and its Industrial Sector? Digital/McKinsey : веб-сайт. URL: https://www.mckinsey.com/~/media/McKinsey/Industries/Semiconductors/Our%20Insights/S

martening%20up%20with%20artificial%20intelligence/Smartening-up-with-artificial-intelligence.ashx

38. Why Machine Learning Is Important For Smarter Manufacturing. towards data science : веб-сайт. URL: https://towardsdatascience.com/why-machine-learning-is-important-for-smarter-manufacturing-27da545de779

39. What is zero trust privilege? Centrify : веб-сайт. URL: https://www.centrify.com/education/what-is-zero-trust-privilege/

40. Top 27 Artificial Neural Network Software. PatResearch : веб-сайт. URL: https://www.predictiveanalyticstoday.com/top-artificial-neural-network-software/

41. An end-to-end open source machine learning platform. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/

42. Best Artificial Neural Network Software. G2 : веб-сайт. URL: https://www.g2.com/categories/artificial-neural-network?utf8=%E2%9C%93&order=g2_score

43. Neural Network Toolbox. Packtx : веб-сайт. URL: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788398435/1/01lvl1sec14/neural-network-toolbox

44. What is TensorFlow? The machine learning library explained. InfoWorld : веб-сайт. URL: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

45. An Open Source Machine Learning Framework for Everyone. GitHub : веб-сайт. URL: https://github.com/tensorflow/tensorflow

46. TensorFlow. ВикипедиЯ : веб-сайт. URL: https://ru.wikipedia.org/wiki/TensorFlow

47. Keras и Tensorflow – классификация текста. КРОК : веб-сайт. URL: https://cloud.croc.ru/blog/byt-v-teme/keras-i-tensorflow-klassifikatsiya-teksta/

48. Top five use cases of TensorFlow. EXASTAX : веб-сайт. URL: https://www.exastax.com/deep-learning/top-five-use-cases-of-tensorflow/

49. Various Uses of TensorFlow. DZone : веб-сайт. URL: https://dzone.com/articles/learn-various-uses-of-tensorflow

50. Module: tf. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf

51. The Sequential model. Keras : веб-сайт. URL: https://keras.io/guides/sequential_model/

52. Input object. Keras : веб-сайт. URL: https://keras.io/api/layers/core_layers/input/

53. Activation layer. Keras : веб-сайт. URL: https://keras.io/api/layers/core_layers/activation/

54. Embedding layer. Keras : веб-сайт. URL: https://keras.io/api/layers/core_layers/embedding/

55. Masking layer. Keras : веб-сайт. URL: https://keras.io/api/layers/core_layers/masking/

56. Lambda layer. Keras : веб-сайт. URL: https://keras.io/api/layers/core_layers/lambda/

57. Dense layer. Keras : веб-сайт. URL: https://keras.io/api/layers/core_layers/dense/

58. Module: tf.keras.initializers. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf/keras/initializers

59. Layer activation functions. Keras : веб-сайт. URL: https://keras.io/api/layers/activations/

60. Layer weight regularizers. Keras : веб-сайт. URL: https://keras.io/api/layers/regularizers/

61. Module: tf.keras.constraints. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf/keras/constraints

62. Regularization. ML Glossary : веб-сайт. URL: https://ml-cheatsheet.readthedocs.io/en/latest/regularization.html

63. Activation Functions. ML Glossary : веб-сайт. URL: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

64. Module: tf.keras.optimizers. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

65. Optimizers. ML Glossary : веб-сайт. URL: https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html

66. An overview of gradient descent optimization algorithms. Sebastian Ruder : веб-сайт. URL: https://ruder.io/optimizing-gradient-descent/index.html

67. Метод сопряжённых градиентов. Распознавание : веб-сайт. URL: http://www.machinelearning.ru/wiki/index.php?title=Метод_сопряжённых_градиентов

68. Метод BFGS или один из самых эффективных методов оптимизации. Пример реализации на Python. Habr : веб-сайт. URL: https://habr.com/ru/post/333356/

69. Convolutional Neural Networks for Visual Recognition. CS231n : веб-сайт. URL: https://cs231n.github.io/neural-networks-3/#sgd

70. The Newton's method. Wikiversity : веб-сайт. URL: https://en.wikiversity.org/wiki/The_Newton%27s_method

71. tf.keras.losses.Loss. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf/keras/losses/Loss

72. Module: tf.keras.metrics. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf/keras/metrics

73. tf.keras.Model. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/api_docs/python/tf/keras/Model

74. API Reference. Scikit learn : веб-сайт. URL: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

75. Overfitting in Machine Learning: What It Is and How to Prevent It. ELITE Data Science : веб-сайт. URL: https://elitedatascience.com/overfitting-in-machine-learning

76. 16. Искусственные нейронные сети. Интеллектуальные информационные системы : веб-сайт. URL: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema16

77. Install TensorFlow with pip. TensorFlow : веб-сайт. URL: https://www.tensorflow.org/install/pip

Додаток А

Текст програми моделювання стійкості твердосплавних різців при обробленні титанових сплавів

```
# import model
import numpy as np
import tensorflow as tf
from sklearn.linear_model import Ridge  #import a model Ridge
from sklearn.linear_model import RidgeCV  #import model RidgeCV
from sklearn.model_selection import GridSearchCV
from scipy.stats import uniform as sp_rand
from sklearn.model_selection import RandomizedSearchCV
from sklearn.linear_model import LinearRegression #import ordinary least squares LinearRegression model

in_train = np.array([[1.0, 0.4, 1.0],
            [0.7,0.4,1.0],
            [1.2, 0.3, 1.0],
            [0.8,0.3,1.0],
            [1.6,0.2,1.0],
            [1.4,0.2,1.0],
            [1.0,0.2,1.0],
            [0.9,0.2,1.0],
            [2.0,0.08,1.0],
            [1.2,0.08,1.0],
            [1.2,0.2,4.0],
            [1.0,0.2,4.0],
            [1.5,0.2,2.0],
            [1.2,0.2,2.0],
            [1.6,0.2,1.0],
            [1.4,0.2,1.0],
```

```
                [0.9,0.2,1.0]], dtype=float)
out_train                                                                                                            =
np.array([1.8,7.5,2.7,8.1,3.0,4.5,10.8,16.2,2.9,14.9,2.5,7.8,2.3,7.0,3.0,4.4,17.1], dtype=float)

in_test = np.array([[0.9, 0.4, 0.1],
                [1.0,0.3,0.1],
                [1.0,0.2,0.2]], dtype=float)
out_test = np.array([4.2,5.4,11.0], dtype=float)

#RidgeCV
# prepare a range of alpha values to test
alphas = np.array([0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0],dtype=float)
# instantiate
linreg=RidgeCV(alphas).fit(in_train, out_train)
r2=linreg.score(in_train,out_train)
print('R^2: {}'.format(r2))

# make predictions on the training set
out_pred1 = linreg.predict(in_train)
print(out_pred1)

print("\nTrain:")
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_train, out_pred1))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_train, out_pred1))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_train, out_pred1)))
print()
```

```
# make predictions on the testing set
out_pred2 = linreg.predict(in_test)
print(out_pred2)

print("\nTest:")
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_test, out_pred2))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_test, out_pred2))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_test, out_pred2)))
print('F =',(r2*(29-3-1))/((1-r2)*(3)))

#Ridge with random
# prepare a uniform distribution to sample for the alpha parameter
param_grid = {'alpha': sp_rand()}
# create and fit a ridge regression model, testing random alpha values
model = Ridge()
rsearch = RandomizedSearchCV(estimator=model, param_distributions=param_grid,
n_iter=100)
rsearch.fit(in_train, out_train)
# summarize the results of the random parameter search
print(rsearch.best_score_)
print(rsearch.best_estimator_.alpha)
r2=rsearch.score(in_train,out_train)
print('R^2: {}'.format(r2))

# make predictions on the training set
out_pred1 = rsearch.predict(in_train)
print(out_pred1)
```

```
print("\nTrain:")
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_train, out_pred1))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_train, out_pred1))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_train, out_pred1)))
print()

# make predictions on the testing set
out_pred2 = rsearch.predict(in_test)
print(out_pred2)

print("\nTest:")
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_test, out_pred2))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_test, out_pred2))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_test, out_pred2)))
print('F =',(r2*(29-3-1))/((1-r2)*(3)))

linreg = LinearRegression()
# fit the model to the training data (learn the coefficients)
linreg.fit(in_train, out_train)
r2=linreg.score(in_train,out_train)
print('R^2: {}'.format(r2))# print the intercept and coefficients
print('Intercept: ', linreg.intercept_)
print('Coefficient: ', linreg.coef_)
```

```
# make predictions on the training set
out_pred1 = linreg.predict(in_train)
print(out_pred1)

print("\nTrain:")
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_train, out_pred1))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_train, out_pred1))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_train, out_pred1)))

# make predictions on the testing set
out_pred2 = linreg.predict(in_test)
print(out_pred2)
print("\nTest:")
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_test, out_pred2))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_test, out_pred2))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_test, out_pred2)))
print('F =',(r2*(29-3-1))/((1-r2)*(3)))

from sklearn.preprocessing import PolynomialFeatures

def create_polynomial_regression_model(degree):
  poly_features = PolynomialFeatures(degree=degree)
  # transforms the existing features to higher degree features.
  X_train_poly = poly_features.fit_transform(in_train)
```

```
# fit the transformed features to Linear Regression
poly_model = LinearRegression()
poly_model.fit(X_train_poly, out_train)

# predicting on training data-set
y_train_predicted = poly_model.predict(X_train_poly)

# predicting on test data-set
y_test_predict = poly_model.predict(poly_features.fit_transform(in_test))

# evaluating the model on training dataset
rmse_train = np.sqrt(tf.metrics.mean_squared_error(out_train, y_train_predicted))
r2_train = tf.metrics.r2_score(out_train, y_train_predicted)

# evaluating the model on test dataset
rmse_test = np.sqrt(tf.metrics.mean_squared_error(out_test, y_test_predict))
r2_test = tf.metrics.r2_score(out_test, y_test_predict)

print("The model performance for the training set")
print("R2 score of training set is {}".format(r2_train))
# calculate MAE
print("MAE =", tf.metrics.mean_absolute_error(out_train, y_train_predicted))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_train, y_train_predicted))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_train, y_train_predicted)))

print("The model performance for the test set")
print("R2 score of test set is {}".format(r2_test))
# calculate MAE
```

```
print("MAE =", tf.metrics.mean_absolute_error(out_test, y_test_predict))
# calculate MSE
print("MSE =", tf.metrics.mean_squared_error(out_test, y_test_predict))
# calculate RMSE
print("RMSE =", np.sqrt(tf.metrics.mean_squared_error(out_test, y_test_predict)))
r4=(r2_train/(1-r2_train))*((29-3-1)/3)
print('F =',r4)
```

Додаток Б

Програмний код моделювання обробки жаростійкої сталі фрезеруванням за різних режимів різання

```
import tensorflow as tf
import numpy as np
model = tf.keras.Sequential()

model.add(tf.keras.layers.Dense(50, input_dim=5, activation='relu', bias_initializer='glorot_normal'))
model.add(tf.keras.layers.Dense(50, input_dim=50, activation='relu', bias_initializer='glorot_normal'))
model.add(tf.keras.layers.Dense(1, input_dim=50, activation='linear', bias_initializer='glorot_normal'))

in_train = np.array([[0.8,0.55,1,1.4,0.35],
        [0.8,0.55,1,1.9,0.32],
        [0.8,0.55,2,0.7,0.36],
        [0.8,0.55,2,1.65,0.275],
        [0.8,0.55,3,0.65,0.33],
        [0.8,0.55,3,1.1,0.28],
        [0.8,0.55,3,1.5,0.25],
        [1.25,0.85,1,0.95,0.33],
        [1.25,0.85,1,1.25,0.295],
        [1.25,0.85,1,1.75,0.27],
        [1.25,0.85,2,0.85,0.29],
        [1.25,0.85,2,1.5,0.235],
        [1.25,0.85,3,0.75,0.265],
        [1.25,0.85,3,1.0,0.24],
        [1.25,0.85,3,1.4,0.215],
        [1.6,1.0,2,0.9,0.25],
```

```
                    [1.6,1.0,2,1.5,0.21],
                    [1.6,1.0,3,0.85,0.235],
                    [1.6,1.0,3,1.15,0.21],
                    [1.6,1.0,3,1.35,0.19],
                    [2.0,1.2,1,1.2,0.25],
                    [2.0,1.2,1,1.35,0.24],
                    [2.0,1.2,1,1.45,0.23],
                    [2.0,1.2,2,1.05,0.22],
                    [2.0,1.2,2,1.28,0.2],
                    [2.0,1.2,3,0.96,0.2],
                    [2.0,1.2,3,1.1,0.19]], dtype=float)
out_train = np.array([0.7, 0.94, 0.7, 1.63, 0.97, 1.63, 2.23, 0.73, 0.96, 1.34, 1.3, 2.3, 1.72,
2.3, 3.22, 1.78, 2.97, 2.52, 3.42, 4.01, 1.3, 1.47, 1.6, 2.1, 2.56, 2.85, 3.1], dtype=float)

in_test = np.array([[0.8,0.55,2,1.2,0.305],
                    [1.25,0.85,2,1.1,0.26],
                    [1.6,1.0,2,1.25,0.23],
                    [2.0,1.2,2,1.2,0.21]], dtype=float)

true = np.array([1.19,1.68,2.48,2.38], dtype=float)

class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end (self,epoch,logs={}):
    if(logs.get('loss')<0.001):
      print("\nReached loss<0.001 so cancelling training!")
      self.model.stop_training = True

callbacks=myCallback()
sgd = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd, loss='mean_squared_error', run_eagerly=True)
```

```
model.summary()
model.fit(in_train, out_train, epochs=1200, callbacks=[callbacks],
validation_data=(in_test, true))

# make predictions on the testing set
out_pred = model.predict(in_train)
print(out_pred)
print(out_train)

from sklearn import metrics

def mean_error(x, y):
  # calculate MAE using scikit-learn
  print("MAE =", metrics.mean_absolute_error(x, y))
  # calculate MSE using scikit-learn
  print("MSE =", metrics.mean_squared_error(x, y))
  # calculate RMSE using scikit-learn
  print("RMSE =", np.sqrt(metrics.mean_squared_error(x, y)))

# make predictions on the testing set
mean_error(out_train, out_pred)
out_pred = model.predict(in_test)
print(out_pred)
mean_error(true, out_pred)
```

Додаток В

Код програми моделювання процесу фрезерування площинних поверхонь заготівок із корозійностійкої сталі торцевими фрезами з пластинами із твердого сплаву без охолодження

```
import tensorflow as tf
import numpy as np
from __future__ import absolute_import, division, print_function, unicode_literals
import os
print(tf.version.VERSION)
# compare scaling methods for mlp inputs on regression problem
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import SGD
from matplotlib import pyplot
from numpy import mean
from numpy import std
from keras.callbacks import ModelCheckpoint
from keras.layers import Dropout
from sklearn import metrics

!pip install -q pyyaml h5py  # Требуется для сохранения модели в формате HDF5

checkpoint_path_1 = "/content/cp_1.ckpt"
checkpoint_dir_1 = os.path.dirname(checkpoint_path_1)

checkpoint_path_2 = "/content/cp_2.ckpt"
checkpoint_dir_2 = os.path.dirname(checkpoint_path_2)
```

```
checkpoint_path_3 = "/content/cp_3.ckpt"
checkpoint_dir_3 = os.path.dirname(checkpoint_path_3)
# prepare dataset with input and output scalers, can be none
def get_dataset(input_scaler, output_scaler):

  trainX = np.array([[100.0, 1.0, 70.0, 210.0, 164.0],
                    [100.0, 1.0, 70.0, 330.0, 129.0],
                    [100.0, 1.0, 70.0, 425.0, 112.0],
                    [100.0, 2.0, 70.0, 170.0, 132.0],
                    [100.0, 2.0, 70.0, 340.0, 90.0],
                    [100.0, 5.0, 70.0, 150.0, 119.0],
                    [100.0, 5.0, 70.0, 236.0, 93.0],
                    [125.0, 1.0, 85.0, 800.0, 124.0],
                    [125.0, 1.0, 85.0, 265.0, 108.0],
                    [125.0, 1.0, 85.0, 300.0, 99.0],
                    [125.0, 2.0, 85.0, 155.0, 99.0],
                    [125.0, 2.0, 85.0, 245.0, 80.0],
                    [125.0, 5.0, 85.0, 145.0, 90.0],
                    [125.0, 5.0, 85.0, 190.0, 78.0],
                    [125.0, 5.0, 85.0, 220.0, 72.0],
                    [160.0, 1.0, 110.0, 190.0, 121.0],
                    [160.0, 2.0, 110.0, 155.0, 97.0],
                    [160.0, 2.0, 110.0, 200.0, 84.0],
                    [160.0, 5.0, 110.0, 140.0, 88.0],
                    [160.0, 5.0, 110.0, 180.0, 77.0],
                    [200.0, 1.0, 120.0, 162.0, 84.0],
                    [200.0, 1.0, 120.0, 210.0, 73.0],
                    [200.0, 2.0, 120.0, 145.0, 76.0],
                    [200.0, 2.0, 120.0, 190.0, 66.0]],dtype = float)
```

```
trainy = np.array([[1.13],
                   [1.78],
                   [2.29],
                   [2.25],
                   [5.50],
                   [4.04],
                   [6.33],
                   [1.31],
                   [1.73],
                   [1.96],
                   [2.94],
                   [4.71],
                   [4.85],
                   [6.22],
                   [7.20],
                   [1.61],
                   [3.94],
                   [5.08],
                   [5.93],
                   [7.62],
                   [4.50],
                   [5.80],
                   [6.50],
                   [8.50]], dtype = float)

testX = np.array([[100.0, 2.0, 70.0, 265.0, 103.0],
                 [125.0, 2.0, 85.0, 210.0, 86.0],
                 [160.0, 2.0, 110.0, 260.0, 73.0],
                 [200.0, 1.0, 120.0, 265.0, 66.0]], dtype = float)
```

```
    testy = np.array([[4.28],
              [4.12],
              [6.61],
              [7.50]], dtype = float)
    # scale inputs
    if input_scaler is not None:
      # fit scaler
      input_scaler.fit(trainX)
      # transform training dataset
      trainX = input_scaler.transform(trainX)
      # transform test dataset
      testX = input_scaler.transform(testX)
    if output_scaler is not None:
      # fit scaler on training dataset
      output_scaler.fit(trainy)
      # transform training dataset
      trainy = output_scaler.transform(trainy)
      # transform test dataset
      testy = output_scaler.transform(testy)
    return trainX, trainy, testX, testy

def create_model():
    # define model
    model = Sequential()
    model.add(Dense(30, use_bias = True, activation='relu', bias_initializer='glorot_normal', kernel_initializer='glorot_uniform', bias_regularizer='l2', kernel_regularizer=tf.keras.regularizers.l1_l2(0.0000000001)))
    model.add(Dense(50, use_bias = True, activation='tanh', bias_initializer='glorot_normal', kernel_initializer='glorot_uniform', bias_regularizer='l2', kernel_regularizer=tf.keras.regularizers.l1_l2(0.0000000001)))
```

```
    model.add(Dense(50, use_bias = True, activation='tanh', bias_initializer='glorot_normal', kernel_initializer='glorot_uniform', bias_regularizer='l2', kernel_regularizer=tf.keras.regularizers.l1_l2(0.0000000001)))
    model.add(Dense(1, use_bias = True, activation='linear', bias_initializer='glorot_normal', kernel_initializer='glorot_uniform', bias_regularizer='l2', kernel_regularizer=tf.keras.regularizers.l1_l2(0.0000000001)))
    # compile model
    optimizer = tf.keras.optimizers.Adamax( learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, name='Adamax' )
    model.compile(loss='mean_squared_error', optimizer=optimizer, run_eagerly=True)
    return model

trainX, trainy, testX, testy = get_dataset(None, None)

scaler_0 = StandardScaler()
scaler_1 = MinMaxScaler()
scaler_2 = StandardScaler()
scaler_3 = StandardScaler()
scaler_4 = StandardScaler()

scaler_0.fit(trainX, trainy)
scaler_1.fit(trainX, trainy)
scaler_2.fit(trainX, trainy)
scaler_3.fit(trainX, trainy)
scaler_4.fit(trainX, trainy)

model = create_model()

# fit and evaluate mse of model on test set
def evaluate_model(trainX, trainy, testX, testy, n):
```

```
  if n == 1:
    cp_callback = ModelCheckpoint(filepath=checkpoint_path_1,
                              save_weights_only=True,
                              verbose=1)
  elif n == 2:
    cp_callback = ModelCheckpoint(filepath=checkpoint_path_2,
                              save_weights_only=True,
                              verbose=1)
  elif n == 3:
    cp_callback = ModelCheckpoint(filepath=checkpoint_path_3,
                              save_weights_only=True,
                              verbose=1)

  # fit model
  model.fit(trainX, trainy, validation_split = 0.33, epochs=1200,
          callbacks=[cp_callback], verbose=0, validation_data=(testX, testy))
  if n == 1:
    !ls {checkpoint_dir_1}
  elif n == 2:
    !ls {checkpoint_dir_2}
  elif n == 3:
    !ls {checkpoint_dir_3}
  # evaluate the model
  test_mse = model.evaluate(testX, testy, verbose=0)
  return test_mse

# evaluate model multiple times with given input and output scalers
def repeated_evaluation(input_scaler, output_scaler, n_repeats=30):
  # get dataset
  trainX, trainy, testX, testy = get_dataset(input_scaler, output_scaler)
```

```
# repeated evaluation of model
if input_scaler is None:
  n = 1
elif input_scaler is scaler_1:
  n = 2
elif input_scaler is scaler_3:
  n = 3

results = list()
for i in range(n_repeats):
  test_mse = evaluate_model(trainX, trainy, testX, testy, n)
  print('>%.3f' % test_mse)
  results.append(test_mse)
  if i ==  n_repeats-1:
    model.summary()
    if input_scaler is None:
      unscaled_1 = model.predict(testX)
      unscaled_2 = scaler_0.inverse_transform(testy)
      print(scaler_0.inverse_transform(model.predict(testX)))
      print(scaler_0.inverse_transform(testy))
    elif input_scaler is scaler_1:
      norm_1 = scaler_2.inverse_transform(model.predict(testX))
      norm_2 = scaler_2.inverse_transform(testy)
      print(scaler_2.inverse_transform(model.predict(testX)))
      print(scaler_2.inverse_transform(testy))
    else:
      scaled_1 = scaler_4.inverse_transform(model.predict(testX))
      scaled_2 = scaler_4.inverse_transform(testy)
      print(scaler_4.inverse_transform(model.predict(testX)))
      print(scaler_4.inverse_transform(testy))
```

```
  return results

def calc_error(testx, testy):
  MAE = metrics.mean_absolute_error(testx, testy)
  MSE = metrics.mean_squared_error(testx, testy)
  RMSE = np.sqrt(MSE)
  print("MAE = ", MAE)
  print("MSE = ", MSE)
  print("RMSE = ", RMSE)

# unscaled inputs
print('unscaled inputs')
results_unscaled_inputs = repeated_evaluation(None, scaler_0)

# summarize results
print('Unscaled: %.3f (%.3f)' % (mean(results_unscaled_inputs), std(results_unscaled_inputs)))

#unscaled
# Создайте экземпляр базовой модели
model = create_model()
trainX, trainy, testX, testy = get_dataset(None, scaler_0)
# Оцените модель
loss = model.evaluate(testX,  testy, verbose=2)
print("Untrained model, loss: {:5.2f}".format(loss))

# Загрузим веса
model.load_weights(checkpoint_path_1)

# Оценим модель снова
```

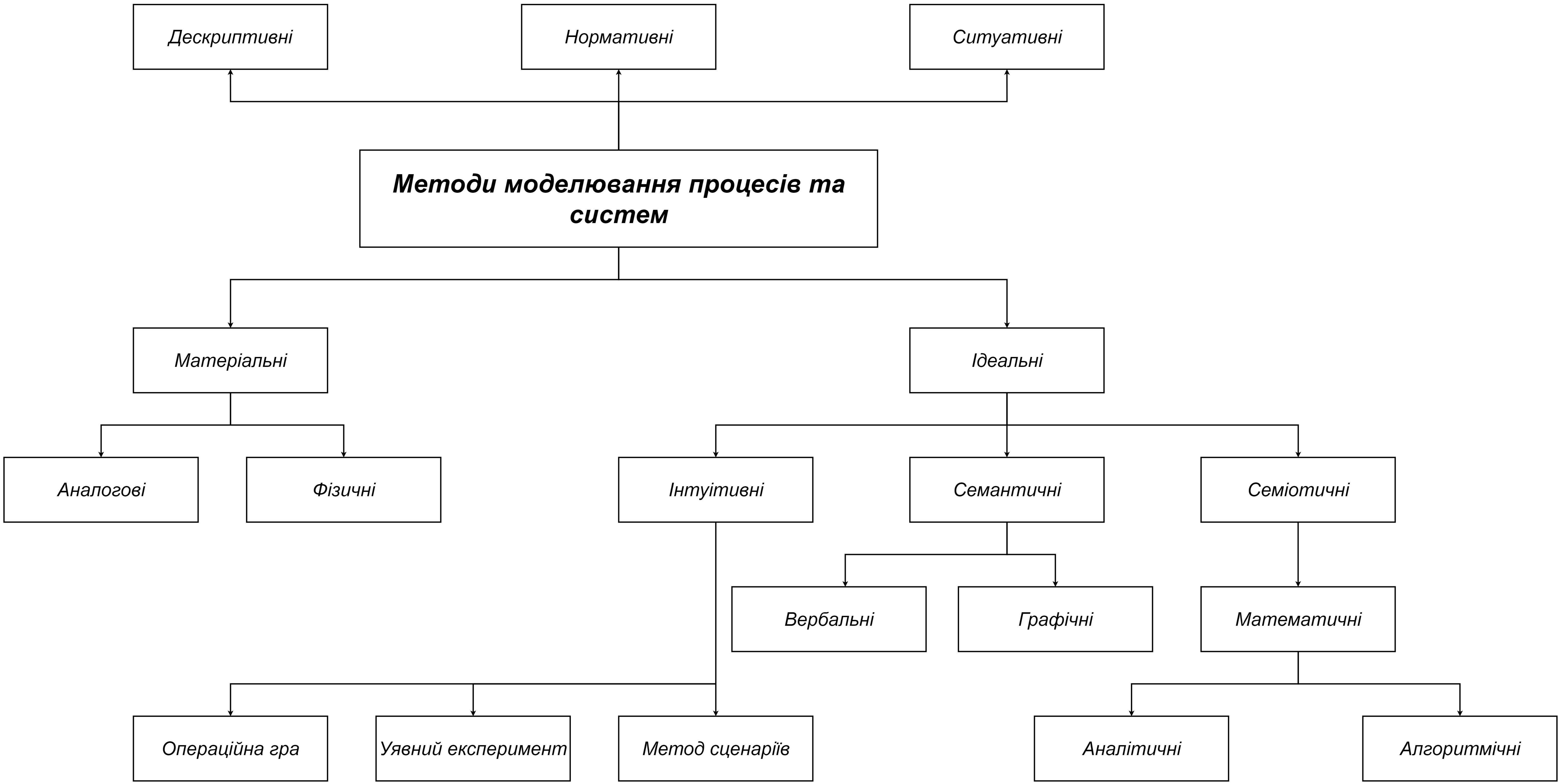
```
loss = model.evaluate(testX,  testy, verbose=2)
print("Restored model, loss: {:5.2f}".format(loss))

trainy_pr = model.predict(trainX)
print(scaler_0.inverse_transform(trainy_pr))
print(scaler_0.inverse_transform(trainy))
calc_error(trainy_pr, trainy)

testy_pr = model.predict(testX)
print(scaler_0.inverse_transform(testy_pr))
print(scaler_0.inverse_transform(testy))
calc_error(testy_pr, testy)

# standardized inputs
print("standardized inputs")
results_standardized_inputs = repeated_evaluation(scaler_3, scaler_4)

print('Standardized: %.3f (%.3f)' % (mean(results_standardized_inputs), std(results_standardized_inputs)))
#standartalized
# Создайте экземпляр базовой модели
model = create_model()
trainX, trainy, testX, testy = get_dataset(scaler_3, scaler_4)
# Оцените модель
loss = model.evaluate(testX,  testy, verbose=2)
print("Untrained model, loss: {:5.2f}".format(loss))

# Загрузим веса
model.load_weights(checkpoint_path_3)
```

```
# Оценим модель снова
loss = model.evaluate(testX,  testy, verbose=2)
print("Restored model, loss: {:5.2f}".format(loss))

trainy_pr = model.predict(trainX)
print(scaler_4.inverse_transform(trainy_pr))
print(scaler_4.inverse_transform(trainy))
calc_error(trainy_pr, trainy)

testy_pr = model.predict(testX)
print(scaler_4.inverse_transform(testy_pr))
print(scaler_4.inverse_transform(testy))
calc_error(scaler_4.inverse_transform(testy_pr), scaler_4.inverse_transform(testy))

# normalized inputs
print("normalized inputs")
results_normalized_inputs = repeated_evaluation(scaler_1, scaler_2)

print('Normalized: %.3f (%.3f)' % (mean(results_normalized_inputs), std(results_normalized_inputs)))

#normalized
# Создайте экземпляр базовой модели
model = create_model()
trainX, trainy, testX, testy = get_dataset(scaler_1, scaler_2)
# Оцените модель
loss = model.evaluate(testX,  testy, verbose=2)
print("Untrained model, loss: {:5.2f}".format(loss))

# Загрузим веса
```

```
model.load_weights(checkpoint_path_2)

# Оценим модель снова
loss = model.evaluate(testX,  testy, verbose=2)
print("Restored model, loss: {:5.2f}".format(loss))

trainy_pr = model.predict(trainX)
print(scaler_2.inverse_transform(trainy_pr))
print(scaler_2.inverse_transform(trainy))
calc_error(trainy_pr, trainy)

testy_pr = model.predict(testX)
print(scaler_2.inverse_transform(testy_pr))
print(scaler_2.inverse_transform(testy))

calc_error(scaler_2.inverse_transform(model.predict(testX)),
scaler_2.inverse_transform(testy))

!pip install -q pyyaml h5py  # Требуется для сохранения модели в формате HDF5

# Сохраним всю модель в  HDF5 файл
model.save('my_model_norm.h5')

# Восстановим в точности ту же модель, включая веса и оптимизатор
new_model = tf.keras.models.load_model('my_model_norm.h5')

# Покажем архитектуру модели
new_model.summary()
new_model.compile(loss='mean_squared_error',
optimizer=tf.keras.optimizers.Adamax(), run_eagerly=True)
```

```
loss = new_model.evaluate(testX,  testy, verbose=2)
print("Точность восстановленной модели: {:5.2f}".format(loss))
```

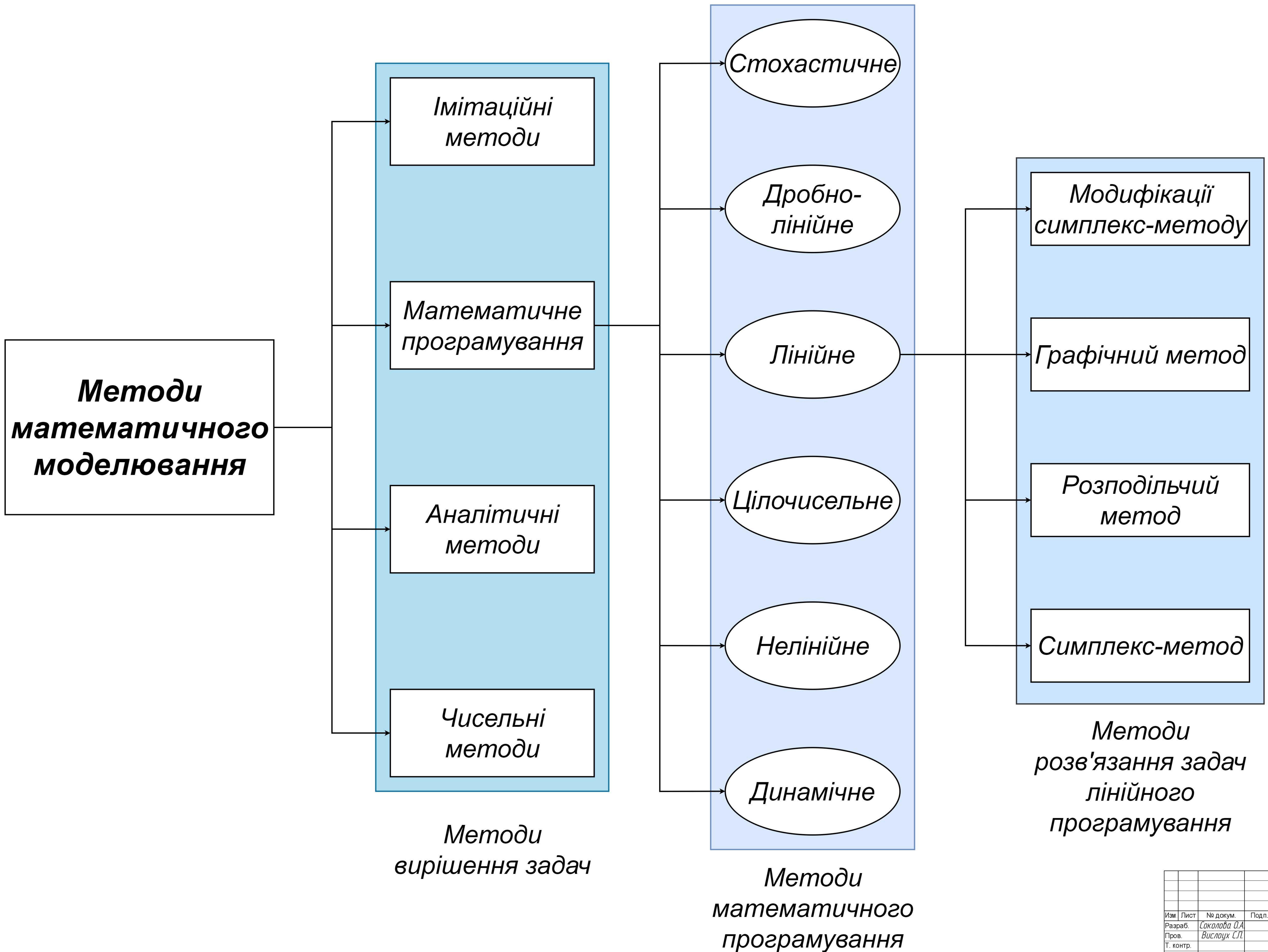
Методи моделювання процесів і систем



					ДР ПБ61.17.1720.001 СХ	
Изм	Лист	№ докум.	Подп.	Дата		
Разраб.		Соколова О.А.			Методи моделювання процесів і систем	Літ. / Масса / Масштаб
Пров.		Вислоух С.П.				
Т. контр.						Лист / Листов
Н. контр.						
Утв.						

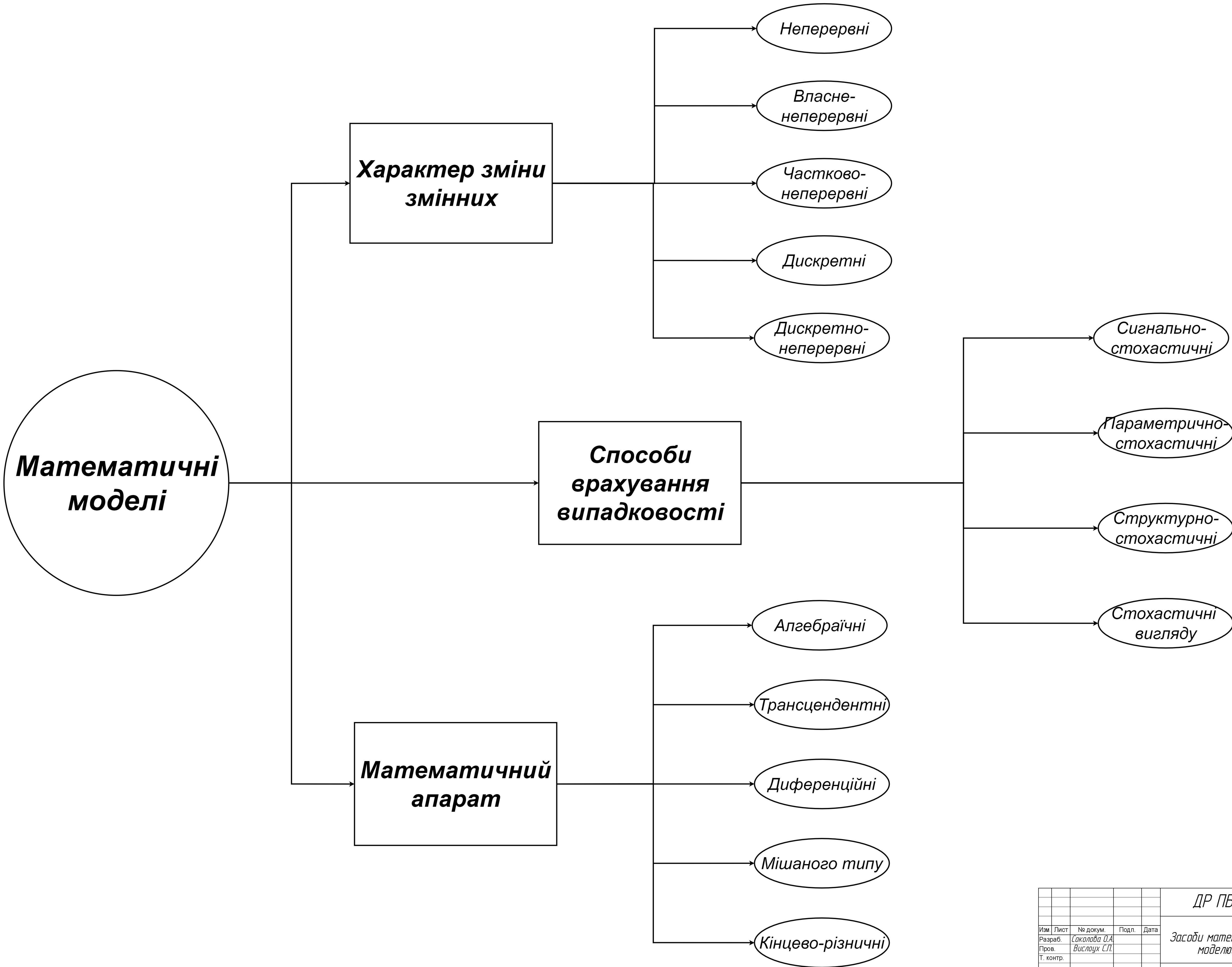
Перв. примен.
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

Методи математичного моделювання технологічних процесів та їх параметрів



Изм	Лист	№ докум.	Подп.	Дата	ДР ПБ61.17.1720.002 СХ			
Разраб.		Соколова О.А.			Методи математичного моделювання	Лит.	Масса	Масштаб
Пров.		Вислоух С.П.						
Т. контр.						Лист	Листов	
Н. контр.								
Утв.								

Засоби математичного моделювання процесів і систем

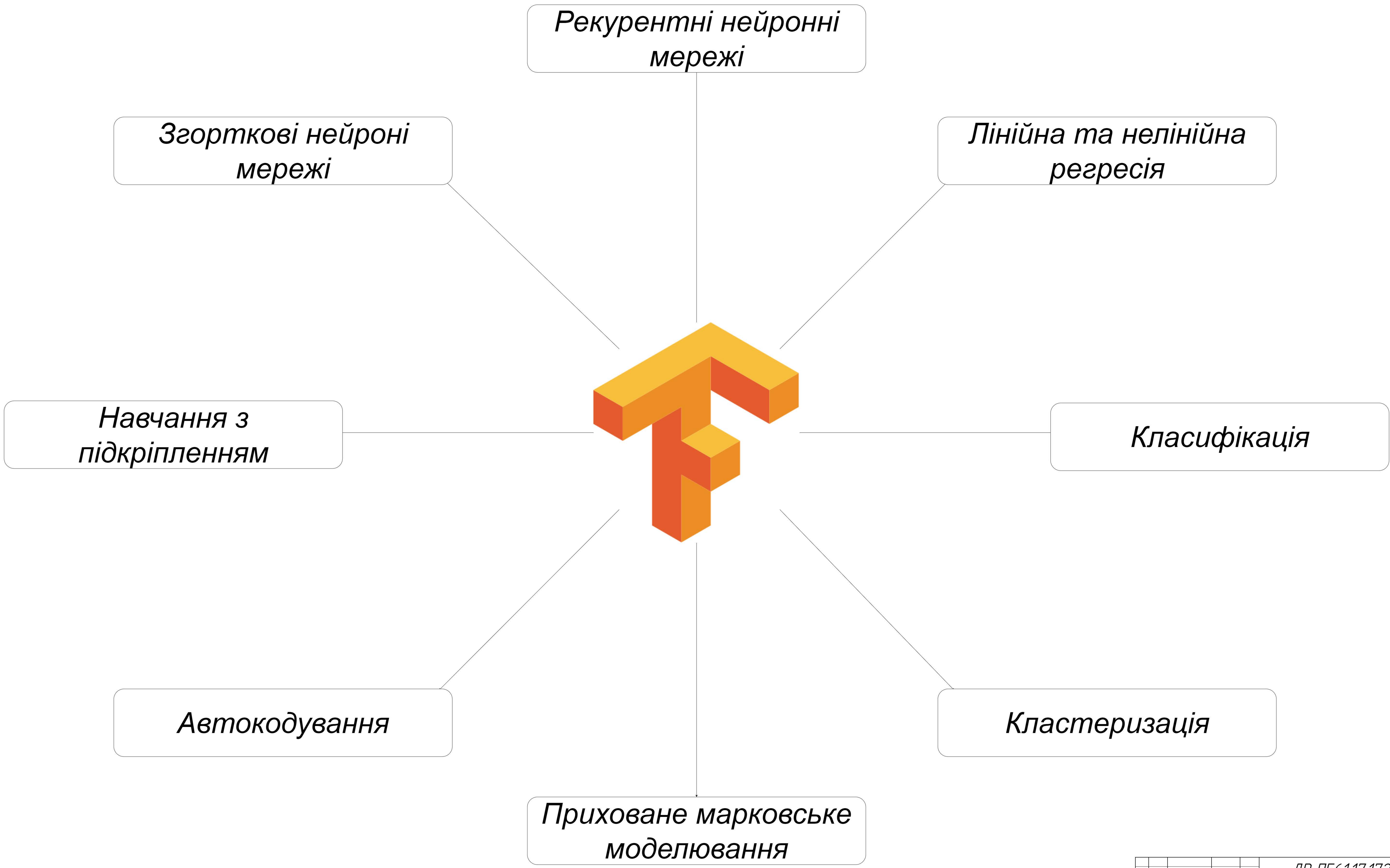


					ДР ПБ61.17.1720.003 СХ			
Изм	Лист	№ докум.	Подп.	Дата		Лит.	Масса	Масштаб
Разраб.		Соколова О.А.			Засоби математичного моделювання			
Пров.		Вислоух С.Л.						
Т. контр.						Лист	Листов	
Н. контр.								
Утв.								

Копировал Формат А1

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Можливості моделювання засобами TensorFlow

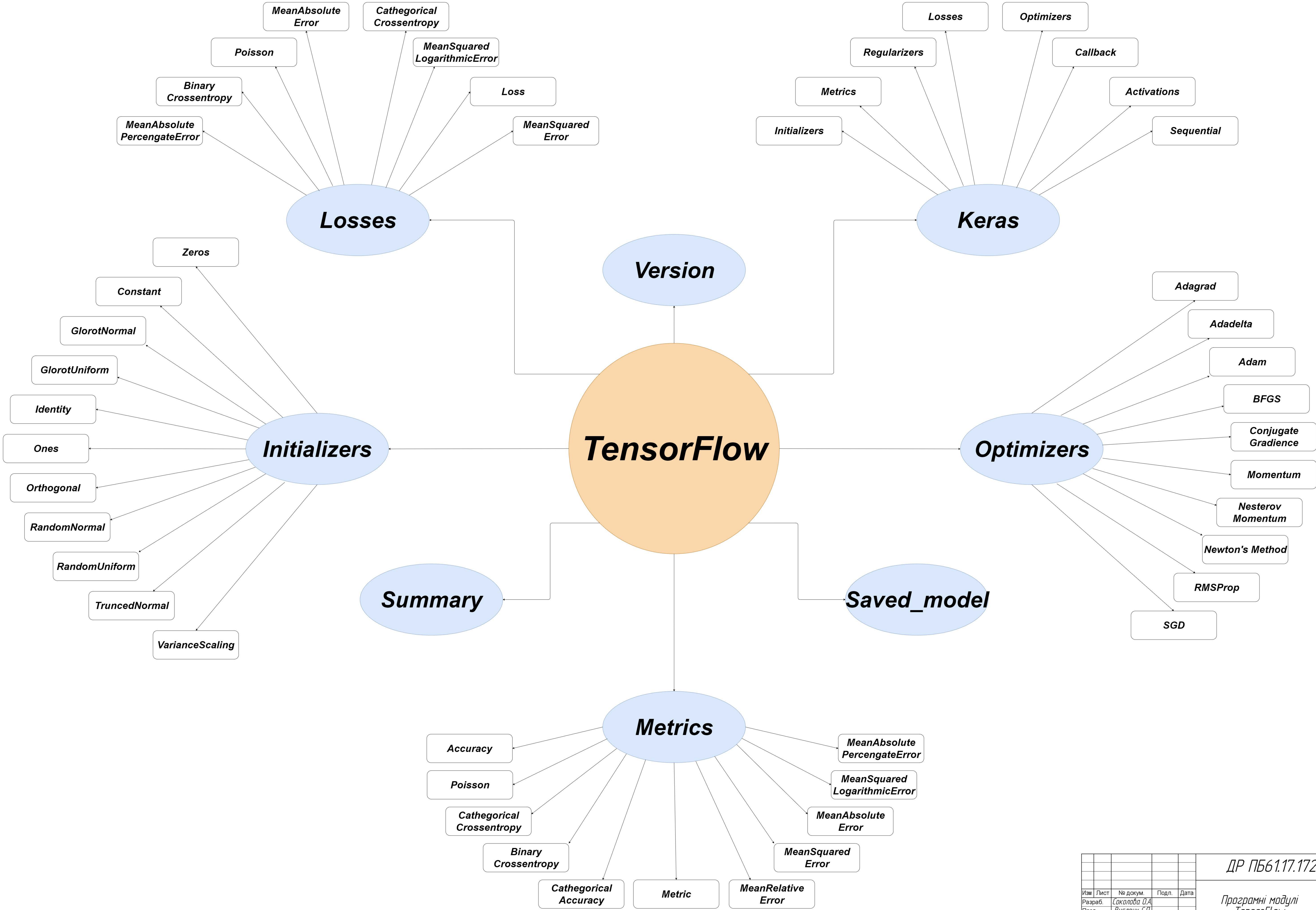


Изм	Лист	№ докум.	Подп.	Дата	ДР ПБ61.17.1720.004 СХ		
Разраб.		Соколова О.А.			Можливості моделювання засобами TensorFlow	Лит.	Масса	Масштаб
Пров.		Вислоух С.П.					
Т. контр.						Лист	Листов
Н. контр.							
Утв.							

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал | Формат А1

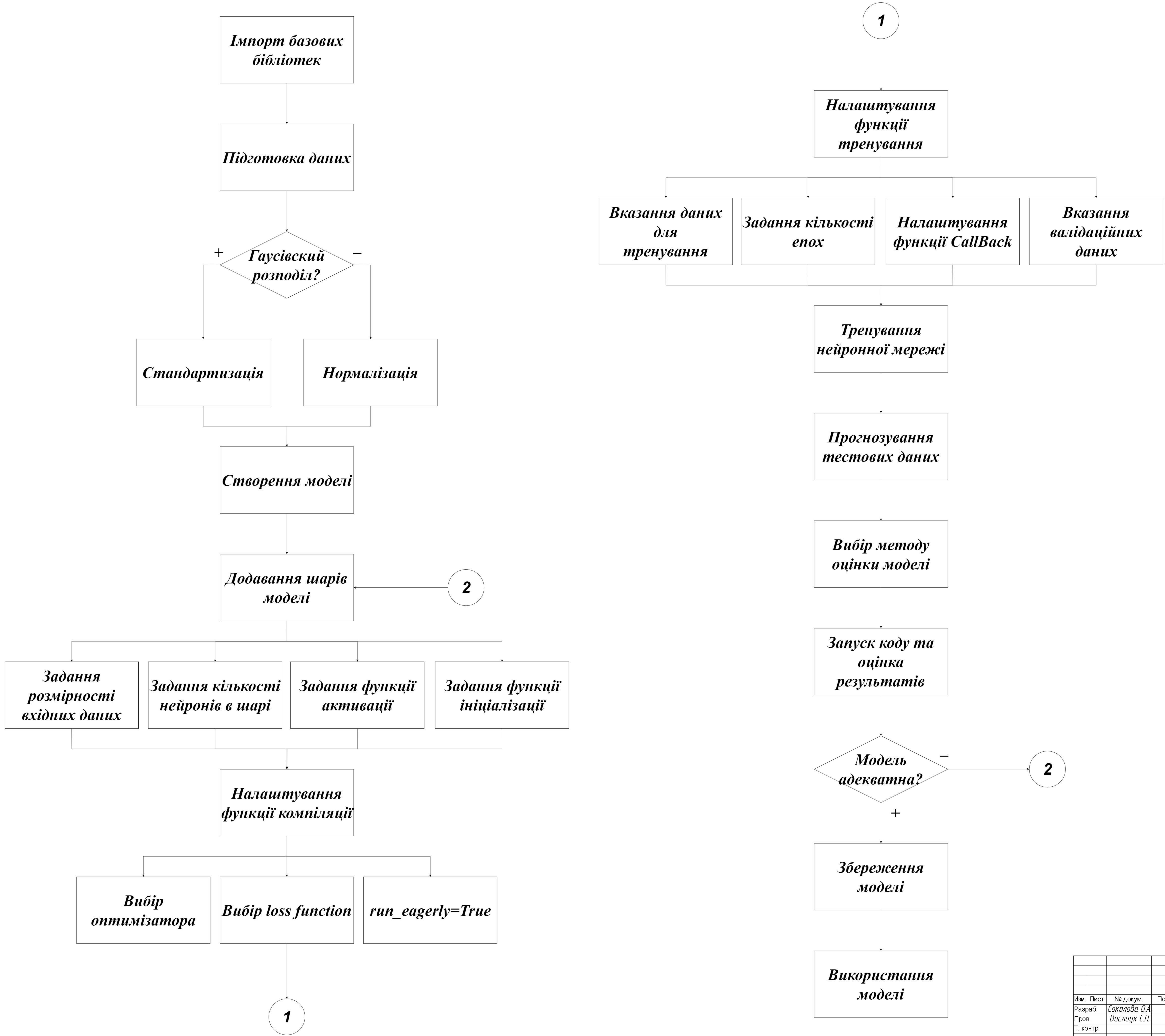
Програмні модулі TensorFlow



Изм	Лист	№ докум.	Подп.	Дата	ДР ПБ61.17.1720.005 СХ
Разраб.		Соколова О.А.			Програмні модулі TensorFlow
Пров.		Вислоух С.П.			Лит. / Масса / Масштаб
Т. контр.					Лист 1 / Листов
Н. контр.					
Утв.					

Перв. примен.
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

Алгоритм моделювання засобами TensorFlow



ДР ПБ61.17.1720.006 СХ					
Изм	Лист	№ докум.	Подп.	Дата	
Разраб.		Соколова О.А.			Алгоритм застосування TensorFlow
Пров.		Вислоух С.П.			
Т. контр.					Лит. / Масса / Масштаб
Н. контр.					Лист / Листов
Утв.					

Копировал Формат А1

Перв. примен. Справ. № Подп. и дата Инв. № дубл. Взам. инв. № Подп. и дата Инв. № подл.

Моделювання технологічних параметрів засобами ШНМ з використанням TensorFlow

Постановка задачі. Виконати моделювання процесу фрезерування площинних поверхонь заготовок з жаростійкої сталі 10Х11Н20Т3Р торцевими фрезами з пластинами із швидкорізальної сталі Р6М5К5 для прогнозування потужності верстату.

Фрагмент вхідних даних

D, мм	B, мм	t, мм	S, мм/хв	V, мм/хв	N, кВт
80	55	1	140	35	0,7
80	55	1	190	32	0,94
80	55	2	70	36	0,7
80	55	3	65	33	0,97
80	55	3	110	28	1,63
125	85	1	95	33	0,73
125	85	1	125	29,5	0,96
125	85	2	150	23,5	2,3
125	85	3	75	26,5	1,72
160	100	2	90	25	1,78
160	100	2	150	21	2,97
160	100	3	85	23,5	2,52
160	100	3	135	19	4,01
200	120	1	135	24	1,47
200	120	2	105	22	2,1
200	120	2	128	20	2,56
200	120	3	110	19	3,1

Результати прогнозування

D, мм	B, мм	t, мм	S, мм/хв	V, мм/хв	N (цільове), кВт	N (прог), кВт
80	55	2	120	30,5	1,19	1.13
125	85	2	110	26	1,68	1.75
160	100	2	125	23	2,48	2.34
200	120	2	120	21	2,38	2.35

Похибка прогнозування

Дані для тренування			Дані для тестування		
MAE	MSE	RMSE	MAE	MSE	RMSE
0.04457	0.00314	0.05606	0.073625	0.00693	0.08329

Постановка задачі. Виконати моделювання процесу фрезерування площинних поверхонь заготовок із корозійностійкої сталі 40Х13 торцевими фрезами з пластинами із твердого сплаву ВК6М без охолодження для прогнозування потужності верстату.

Фрагмент вхідних даних

D, мм	B, мм	t, мм	V, об/хв	S, мм/об	N, кВт
100	70	1	164	210	1,13
100	70	1	129	330	1,78
100	70	2	132	170	2,25
100	70	2	90	340	5,5
100	70	5	119	150	4,04
125	85	1	124	800	1,31
125	85	2	99	155	2,94
125	85	2	80	245	4,71
125	85	5	90	145	4,85
125	85	5	78	190	6,22
160	110	1	121	190	1,61
160	110	2	84	200	5,08
160	110	5	88	140	5,93
160	110	5	77	180	7,62
200	120	1	73	210	5,8
200	120	2	66	190	8,5

Результати прогнозування

D, мм	B, мм	t, мм	V, об/хв	S, мм/об	N, кВт	Unscaled	Normalized	Scaled
100	70	2	103	265	4,28	3,951	3,830	3,627
125	85	2	86	210	4,12	3,828	4,149	4,171
160	110	2	73	260	6,61	7,049	6,449	6,733
200	120	1	66	265	7,5	5,314	6,892	7,126

Похибка прогнозування

Похибка	Дані для тренування			Дані для тестування		
	Unscaled	Normalized	Scaled	Unscaled	Normalized	Scaled
MAE	0,01864	0,07818	0,05538	0,81137	0,31195	0,30026
MSE	0,00067	0,01075	0,00535	1,29054	0,14959	0,14570
RMSE	0,02596	0,10366	0,07317	1,13602	0,38677	0,38171

Постановка задачі. Виконати моделювання процесу обробки заготовок з титанового сплаву ВТ6 різцями, оснащених плстинами з твердого сплаву ВК4, для прогнозування періоду стійкості різця

Фрагмент вхідних даних

V, м/хв	S, мм/хв	t, мм	T, хв
100	0.4	0.1	1.8
70	0.4	0.1	7.5
120	0.3	1.0	2.7
80	0.3	1.0	8.1
160	0.2	1.0	3.0
140	0.2	1.0	4.5
100	0.2	1.0	10.8
90	0.2	1.0	16.2
200	0.08	1.0	2.9
120	0.08	1.0	14.9
120	0.2	4.0	2.5
100	0.2	4.0	7.8
150	0.2	2.0	2.3
120	0.2	2.0	7.0
160	0.2	1.0	3.0
140	0.2	1.0	4.4
90	0.2	1.0	17.1

Результати прогнозування

V, м/хв	S, мм/хв	t, мм	T (цільове), хв	T (прогн), хв
90	0,4	0,1	4,2	3.64
100	0,3	1,0	5,4	6.37
100	0,2	2,0	11,0	10.66

ДР ПБ61.17.1720.007 СХ

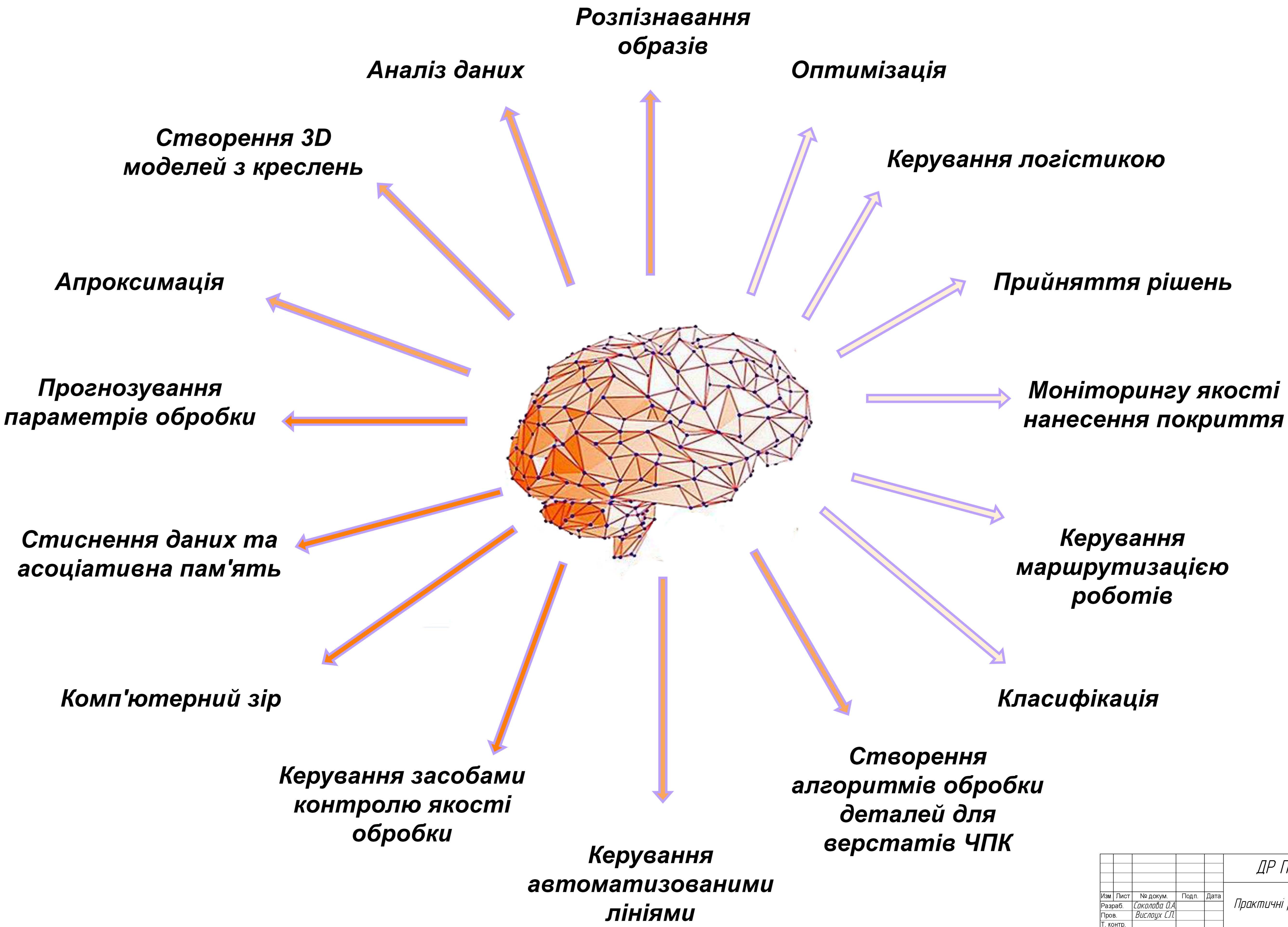
Изм	Лист	№ докум.	Подп.	Дата
Разраб.		Соколова О.А.		
Пров.		Вислоух С.П.		
Т. контр.				
Н. контр.				
Утв.				

Практичне моделювання засобами TensorFlow

Лит. | Масса | Масштаб | Лист | Листов

Копировал | Формат А1

Практичні рекомендації з використання ШНМ в приладобудуванні



					ДР ПБ61.17.1720.008 СХ		
Изм	Лист	№ докум.	Подп.	Дата			
Разраб.		Соколова О.А.			Практичні рекомендації	Літ. Масса Масштаб	
Пров.		Вислоух С.П.					
Т. контр.						Лист	Листов
Н. контр.							
Утв.							

Копировал Формат А1

Перв. примен. Справ. № Подп. и дата Инв. № дубл. Взам. инв. № Подп. и дата Инв. № подл.

ДР ПБ61.17.1720.008 СХ

Висновки

1. Проаналізовано методи моделювання процесів та систем. Встановлено, що найращим методом для моделювання технологічних процесів, є математичне моделювання.
2. Огляд методів математичного моделювання встановив, що найбільш ефективним та перспективним методом математичного моделювання процесів і систем є використання штучних нейронних мереж.
3. Виконано огляд області використання ШНМ та проаналізовано програмні засоби та інструментарій для моделювання за допомогою ШНМ. Сформульовані вимоги до програмних засобів дозволили обрати найбільш раціональний та зручний - платформу TensorFlow.
4. Розглянуто платформу TensorFlow та її області використання. Виконано огляд програмних модулів TensorFlow. Створено алгоритм моделювання процесів для прогнозування параметрів за допомогою TensorFlow.
5. Виконано практичне застосування платформи TensorFlow для створення штучних нейронних мереж для прогнозування технологічних параметрів.
6. Отримані моделі можуть бути успішно впроваджені при проектуванні технологічних процесів.