МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ**

Факультет Комп'ютерних наук, управління та адміністрування

Кафедра Інформаційних технологій

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розпізнавання рукописного тексту з використанням засобів машинного навчання

Виконав студент 2 курсу групи МІС-20
спеціальності 122 Комп'ютерні науки

Бєлодонов Олександр Сергійович

Керівник д.т.н., професор
Мещеряков Володимир Іванович

Рецензент д.т.н., професор
Казакова Надія Феліксівна

Одеса 2021

# АНОТАЦІЯ

на магістерську кваліфікаційну роботу
«Розпізнавання рукописного тексту з використанням засобів машинного навчання»,
студента Бєлодонова Олександра Сергійовича

В сучасному світі при обробці даних часто застосовують алгоритми машинного навчання. У даній магістерський роботі будуть вивчені інструменти, які надають можливість розпізнавати графічні образи, а саме рукописні символи за допомогою інструментів машинного навчання та застосованняна практичному прикладу, а саме аналіз бази даних MNIST.

Мета роботи полягає в аналізі інструментів машинного навчання, вибору більш валідного під мету завдання та його застосування на вибірці даних. Таким чином необхідно виконати такі кроки:

- Завантажити вибірку даних з необхідного джерела
- Підготувати та розрахувати модель навчання
- Запустити тренування моделі на виборці даних
- Провалідувати результати

Валідну модель можна інтегрувати у прилади для подальшого використовування. Схожі моделі або алгоритми можна перевикористовувати у інших проблемних середовищах.

Моделі для розпізнавання рукописного тексту можна використовувати при оцифровуванні паперових документів, розпізнаванні знаків дорожнього руху або номерів машин.

Магістерська робота викладена на 74 сторінках, включає 4 таблиці, 19 рисунків, 3 додатки та 17 літературних посилань.

КЛЮЧОВІ СЛОВА: машинне навчання, нейрона мережа, статистика, розпізнавання.

# SUMMARY

for a master's degree

"Handwriting Recognition using Machine Learning Tools", students of Belodonov Oleksandr

In today’s world, machine learning algorithms are often used in data processing. In this master’s thesis will be studied tools that provide the ability to recognize graphic images, namely handwritten symbols using machine learning tools and applications on a practical example, namely the analysis of the MNIST database.

The purpose of the work is to analyze machine learning tools, select a more valid tools and apply it to the data sample. Thus, following steps must be performed:

- Download a sample of data from the desired source.
- Prepare and calculate a learning model.
- Start model training on the data sample.
- Validate results.

The valid model can be integrated into devices for further use. Similar models or algorithms can be reused in other problem environments.

Handwriting recognition models can be used to digitize paper documents, recognize traffic signs or license plates.

The master’s thesis is explained on 74 pages, includes 4 tables, 19 figures, 3 appendices and 17 references.

KEY WORDS: machine learning, neural network, statistics, recognition.

**ЗМІСТ**

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ.....8
ВСТУП.....9
1 ОПИС ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....11
1.1 Постановка завдання.....11
1.2 Інструменти машинного навчання.....11
1.2 Обґрунтування застосування мови програмування Python.....13
1.3 Обґрунтування вибору інтегрованого середовища PyCharm.....14
1.4 Набір ввідних даних.....14
2 ДОСЛІДЖЕННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ.....16
2.1 Аналіз розвитку нейронних мереж.....16
2.2 Використання фундаментальних складових нейронної мережі.....17
2.3 Навчання нейронної мережі.....24
2.4 Згорткові і багатошарові нейронні мережі.....30
2.4.1 Багатошаровий персептрон.....30
2.4.2 Когнітрон і неокогнітрон Фукушіма.....32
2.4.3 Згорткові нейронні мережі.....33
3 АНАЛІЗ СТАТИСТИЧНИХ ТА МАТЕМАТИЧНИХ МЕТОДІВ.....38
3.1 Використання статистики у контексті нейронних мереж.....38
3.2 Чисельні методи у контексті нейроних мереж. Кореляція.....41
3.3 Стандартне відхилення (девіація) у контексті нейронних мереж.....42
3.4 Використання чисельних методів у контексті обробки та підготовки даних.....44
4 ВИРІШЕННЯ ПРАКТИЧНИХ ЗАВДАНЬ.....47
4.1 Інструментарій побудови нейронних мереж і збору статистики.....47
4.2 Набори символів, які досліджувалися.....48
4.3 Види нейронних мереж, які досліджувалися.....53
4.4 Навчання нейронних мереж та аналіз статистики їх роботи.....59

ВИСНОВКИ.....67
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....69
Додаток А.....71
ВМІСТ ПРОГРАМНОГО МОДУЛЯ MNIST.PY. НАВЧАННЯ НЕЙРОНОЇ МЕРЕЖІ. ОСНОВНА ЧАСТИНА.....71
Додаток Б.....77
ВМІСТ ПРОГРАМНОГО МОДУЛЯ CNN_MNIST.PY. ОЦІНЮВАЧ ЗГОРТКОВИХ НЕЙРОНИХ МЕРЕЖ.....77
Додаток В.....82
ВИКОРИСТАННЯ МОДУЛІВ CNN_MNIST.PY ТА MNIST.PY.....82

## СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

API – Application Programming Interface. Опис методів чи інструментів за допомогою яких різні програми можуть взаємодіяти з іншими.

AWS – Amazon Web Services, набір хмархних технологій від Amazon.

CNN – Convolutional Neural Network Estimator, Оцінювач згорткових нейронних мереж.

CUDA – Compute Unified Device Architecture. Програмно-апаратна архітектура паралельних розрахунків, яка дозволяє збільшити розрахункову потужність завдяки використанню графічних процесорів NVIDIA.

MNIST – Modified National Institute of Standards and Technology, велика база даних рукописних символів.

GCP – Google Cloud Platform, набір хмарних технологій від Google.

MSE – Measure Square Error, квадратична функція помилки.

ReLU – Rectifier linear unit, урізана лінійна функція.

TF — Tensorflow. Інструмент для побудови нейронних мереж.

ЗНР – згорткова нейрона мережа.

# ВСТУП

Історично математика як наука виникла у зв'язку з необхідністю вирішення практичних завдань: будівництво, вимірювань на місцевості, топографія та картографія ,навігації, тощо. У зв’язку з цим математика була розрахунковою, її метою було отримання результату у вигляді числа.

Завдання розв'язку рівнянь постійно виникають на практиці. Наприклад, в економіці, розвиваючи бізнес, ви хочете дізнатися, коли прибуток досягне певного значення, а в медицині при дослідженні дії лікарських препаратів, важливо знати, коли концентрація речовини досягне заданого рівня, тощо. У завданнях оптимізації часто необхідно визначати точки, в яких похідна функції стає рівною нулю, що є необхідною умовою локального екстремуму. У статистиці при побудові оцінок методом найменших квадратів або методом максимальної правдоподібності також доводиться вирішувати нелінійні рівняння та системи рівнянь.

На даний момент існує велика кількість програмних пакетів, які призначені для вирішення запрограмованих або введених користувачем рівнянь. Робота в таких системах вимагає дуже специфічних знань і підготовки. Зараз все частіше постає питання про використання таких систем людьми без підготовки або розрахунок «з листа», тобто постає питання про розпізнавання рукописних або друкованих математичних формул.

Так в процесі роботи над програмою для мобільного телефону, яка може за допомогою камери формувати рівняння, а потім «вирішувати» його з допомогу одного з методів чисельного рішення (наприклад, методом поділу відрізку навпіл), з'явилася проблема розпізнавання графічного відображення математичної формули (її рукописного запису).

На сьогоднішній момент існує велика кількість методів класифікації введених графічних букв і цифр: нейронні мережі (мережі глибинного ма-

шинного навчання, системи штучного інтелекту), метод векторного квантування, метод опорних векторів і багато іншого.

В даний час машинне навчання і нейронні мережі (ланцюги) є актуальними темами в обробці даних.

Завданням даної роботи є вивчення нейронних мереж (ланцюгів), їх типів та методів їх реалізації, а також перевірка на практиці їх роботи на різних вибірках для навчання і тестування. Таким чином, необхідно зібрати дані і виконати дослідження появи помилок під час роботи різноманітних нейронних ланцюгів і спробувати визначити рекомендації по тренуванню і типу використання нейронних мереж для розпізнавання рукописних цифр і букв при розв’язуванні рівнянь.

Завданнями роботи є:

– опрацювати теоретичні матеріали про різні види нейронних мереж;

– побудова та програмна реалізація кількох видів нейронних мереж;

– тренування і тестування нейронних мереж на вибірках MNIST, EMNIST та своїй підготовленій вибірці;

– обробка отриманих результатів з метою визначення максимально відповідної нейронної мережі та її тренування і тестування для задачі розпізнавання математичних нелінійних рівнянь.

# 1 ОПИС ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Постановка завдання

У даному розділі будуть детально розглянути проблеми, завдання, короткий опис потенційних інструментів, які необхідні виконати у даному проекті.

У даному проекті буде розглянута проблема розпізнавання рукописних символів за допомогою машинного навчання. Наразі розпізнавання рукописних символів використовується у повсякденні: від зовнішнього незалежного тестування, де за допомогою алгоритмів розпізнаються символи. Переважно це рукописні цифри. Також дана технологія використовуються для автоматичного заповнення документів та оцифровування документів.

Сама по собі технологія машинного навчання є комплексною та актуальною у наші часи. Поступово аналітики та програмісти розробляють бібліотеки для того, щоб спростити експлуатацію цієї технології. Програмісту або інженеру буде необхідно лише завантажити вибірку даних та за допомогою налаштувань запустити навчання. Даних підхід поступово розробляється у рамках Google Cloud Platform. Даний продукт має систему AutoML, який за допомогою мастера можна налаштувати та використовувати. Наразі даний продукт працює лише з категоризацією та прогнозуванням з задовільною точністю.

## 1.2 Інструменти машинного навчання

Наразі існують такі інструменти для маніпуляцій над нейроними мережами:

Tensorflow – це бібліотека з відкритим вихідним кодом, яка підтримується Google. Має адаптери і бібліотеки у Python

ML.NET – це бібліотека від Microsoft, має алгоритми машинного навчання. Підтримується у середовищі .Net / C#

Google Cloud Platform, AutoML – це хмарний сервіс, який дозволяє оброблювати масиви даних та виконувати прогнозування.

У даному проекті буде використовуватись комбінація Tensoflow з мовою програмування Python.

Python відмінно підходить для невеликих проектів та навчальних чи дослідницьких робіт. Бібліотека Tensorflow має можливості розпізнавання графічних образів, як і ML.NET, але на ML.NET орієнтований для інтеграції у комерційні середовища, які потребують обслуговування та підтримки. Таким чином можна робити дослідження у середовищі Python та TensorFlow посеред аналітиків чи спеціалістів машинного навчання, а потім інтегрувати відлагоджене рішення у комерційних продукт за допомогою програмістів. Також ML.NET має можливість для роботи з Tensorflow.

Google Cloud Platform, AutoML було відкинуто, тому що даний продукт відносно новий та немає підтримки графічної інформації. Але після налаштування та відлагоджування можна опублікувати рішення усередені Google Cloud Platform.

Для роботи з бібліотекою TenforFlow у середовищі Python необхідно використовувати інтегроване середовище розробки. У даному випадку буде використовуватись середовище PyCharm від JetBrains. Даний продукт відмінно підходить для тих, кто комбінує різні інструменті, наприклад Java, C#, Javascript. JetBrains має продукти для кожних з цих технологій, які переважно відрізняються у дрібницях, таким чином не є потреба адаптуватися до іншого середовища.

TenforFlow буде розглянутий більш детальніше пізніше у відповідному розділі.

## 1.2 Обґрунтування застосування мови програмування Python

Python [2] — це високорівнева мова програмування загального призначення з динамічної строгою типізацією і автоматичним управлінням пам'яттю, орієнтований на підвищення продуктивності розробника, читання коду і його якості, а також забезпечення переносимості написаних на ньому програм. Мова є повністю об'єктно-орієнтованим - все є об'єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, за рахунок чого рідко виникає необхідність звернення до документації. Сам же мова відома як інтерпретується і використовується в тому числі для написання скриптів. Недоліками мови є більш низька швидкість роботи і високе споживання енергії написаними на ньому програмах по схожим кодом, написаним на компільованих мовах, як Сі або С ++. Python є мультипарадигмальності мовою програмування, що підтримує імперативне, процедурне, структурний, об'єктно-орієнтоване програмування, метапрограмування і функціональне програмування. Завдання узагальненого програмування вирішуються за рахунок динамічної типізації. Аспектно-орієнтоване програмування підтримується через декоратори, більш повноцінна підтримка додатковими фреймворками. Такі методики як контрактне і логічне програмування можна реалізувати за допомогою бібліотек або розширень. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень з глобальної блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, об'єднуються в пакети.

## 1.3 Обґрунтування вибору інтегрованого середовища PyCharm

PyCharm [3] - інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA. PyCharm - це крос-платформна середовище розробки, яка сумісна з Windows, macOS, Linux. PyCharm Community Edition (безкоштовна версія) знаходиться під ліцензією Apache License, а PyCharm Professional Edition (платна версія) є пропрієтарним ПО.

Має такі відмінності:

- Відладка за допомогою PyDev.
- Автоматичний рефакторинг коду та підказки щодо рефакторингу.
- Підтримка різних систем контроля версій
- Автодоповнення коду

## 1.4 Набір ввідних даних

Для виконання навчання нейронної мережі необхідно мати великій набір даних. У даному випадку це набір зображень рукописних символів та їх розшифровка. У даному проекті було використано набір даних від MNIST [3]. База даних є стандартом, запропонованим Національним інститутом стандартів і технологій США з метою калібрації і зіставлення методів розпізнавання зображень за допомогою машинного навчання в першу чергу на основі нейронних мереж. Дані складаються з заздалегідь підготовлених прикладів зображень, на основі яких проводиться навчання та тестування систем. База даних була створена після переробки оригінального набору чорно-білих зразків розміром 20x20 пікселів NIST. Творці бази даних NIST, в свою чергу, використовували набір зразків з Бюро перепису населення США, до якого бу-

ли додані ще тестові зразки, написані студентами американських університетів. Зразки з набору NIST були нормалізувати, пройшли згладжування і приведені до сірого півтонування розміром 28x28 пікселів.

# 2 ДОСЛІДЖЕННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

## 2.1 Аналіз розвитку нейронних мереж

Штучні нейронні мережі [1] один з найстаріших алгоритмів машинного навчання. Сучасна ера нейронних мереж почалася з новаторською роботи Уоррена Мак-Каллока, психіатра і нейроанатома, і Уолтера Піттца математика. У своїй роботі (1942) вони описали логіку обчислень в нейронних мережах і показали, що мережа, складена з великої кількості елементарних процесорних одиниць, з'єднаних сінаптичними зв'язками, принципово здатна виконати будь-які обчислення.

Історично першою штучною нейронною мережею, здатною до перцепції (сприйняття) і формування реакції на сприйнятий стимул, з'явився Perceptron Розенблатта (Frank Rosenblatt, 1957). Термін "Perceptron" походить від латинського «perceptio», що означає сприйняття, пізнавання. Аналогом цього терміна є "Персептрон". Його автором персептрон розглядався не як конкретний технічний обчислювальний пристрій, а як модель роботи мозку.

Найпростіший класичний персептрон містить елементи трьох типів (мал. 1 та 2 а), призначення яких в цілому відповідає нейрону рефлекторної нейронної мережі Мак-Каллока – Піттца.

У 1956 році Джон фон Нейман (*John von Neumann*) вирішив задачу побудови надійних мереж з нейронів, за допомогою ідеї надмірності, яка була запропонована Террі Алленом Виноградом (*Terry Allen Winograd*) і Кованом (*Cowan J.D.)*.

У 1960-х роках був описаний алгоритм найменших квадратів LMS, який використовувався для побудови адаптивних лінійних елементів Adaline. Для адаптивної класифікації був використаний стохастичний градієнтний метод (1967). У 1965 році вийшла книга Нільса Нільсона (*Nils J. Nilsson*) «Learning Machines». До кінця 1960-х було сформовано багато ідей і конце-

пцій, покладених в основу рекурентних мереж, які отримали назву мереж Хопфілда.

Однак багато проблем були вирішені тільки в 1980-х. Стівен Гросберг відкрив принцип самоорганізації, який отримав назву теорії адаптивного резонансу, який був використаний в мережах Хопфілда. Джон Джозеф Хопфілд показав, що симетричні синаптичні зв'язки гарантують збіжність до стійкого стану. У 1986 році був розроблений алгоритм зворотного поширення помилки, який використовується для навчання нейронних мереж. У 1988 році Брумхед і Лове описали процедуру побудови багатошарової мережі прямого поширення на базі радіальних базисних функцій.

## 2.2 Використання фундаментальних складових нейронної мережі

Нейрон – це обчислювальна одиниця, що отримує інформацію, виконує з нею прості обчислення і передає її далі.

У нейрона є якась обмежена кількість входів; до кожного з цих входів прив'язано деяку вагу і нейрон просто бере і виконує зважену суму своїх входів. Як правило, діапазон значень з якими працює нейрон це [0; 1] або [-1; 1].

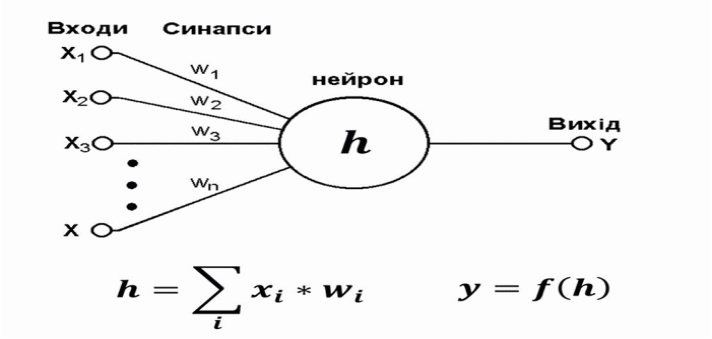


Рисунок 1 — Формальний нейрон

Нейрони діляться на три основні типи: вхідний (н11-н1n), прихований (н21-н2n) і вихідний (н31-н3n) (мал. 3). Також є нейрон зміщення і контекстний нейрон. У тому випадку, коли нейромережа складається з великої кількості нейронів, вводять термін шару. Відповідно, є вхідний шар, який отримує інформацію, n прихованих шарів (зазвичай їх не більше трьох), які її обробляють, і вихідний шар, який виводить результат. У кожного з нейронів є два основні параметри: вхідні дані (input data) і вихідні дані (output data). У разі вхідного нейрона: input = output. В інших – в поле input потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого вона нормалізується за допомогою функції активації (*f(x)*) і потрапляє в поле output.

Синапс – це зв'язок між двома нейронами. У синапсів є один параметр – вага. Завдяки йому вхідна інформація змінюється, коли передається від одного нейрона до іншого. Припустимо, є три нейрона, які передають інформацію наступному нейрону. Тоді у нас є три ваги, відповідні кожному з цих нейронів. У того нейрона, у якого вага буде більше, та інформація і буде домінуючою в наступному нейроні. Насправді, сукупність ваг нейронної мережі або матриця ваг – це своєрідний мозок всієї системи. Саме завдяки цим вагам, вхідна інформація обробляється і перетворюється в результат.

З формул (1) і (2) видно, що вихідна інформація – це значення функції активації (нормалізації) від суми всіх вхідних даних, помножених на відповідні їм ваги. Тобто, по суті, в практичному вигляді відбувається множення матриці вхідних значень на матрицю ваг, а потім виконується активація нейрону (нормалізація).

$$f_1(x) = x_1 * w_1 + x_2 * w_2 + \ldots + x_n * w_n, \qquad (1)$$

Де $X_1$, $X_n$, – значення нейрону, умовні одиниці. $W_1$, $W_n$ – ваги для поточного нейрону X, умовні одиниці.

Якщо, припустимо, в двомірному просторі у нас є деяка множина точок двох класів, а це їх ознаки X1 і X2, тобто, підібравши ці ваги W1 і W2, ми можемо побудувати поверхню, що розділяє їх в цьому просторі. I, таким чином, якщо у нас ця сума, наприклад, більше нуля, то об'єкт належить до першого класу. Якщо ця сума менше нуля, то об'єкт відноситься до другого класу.

$$F_1 = F_{активации}(f_1(x)) \tag{2}$$

***Функція активації*** – це спосіб нормалізації вхідних даних. Тобто, якщо на вході у вас буде велике число, то, пропустивши його через функцію активації, отримаємо вихід в потрібному діапазоні. Функцій активації досить багато, тому розглянемо найосновніші: лінійна, сигмоїд (логістична) і гіперболічний тангенс. Головні їх відмінності – це діапазон значень.

$$f(x) = x \tag{3}$$

Лінійна функція, формула 3, рисунок 2. Ця функція майже ніколи не використовується; за винятком випадків, коли потрібно протестувати нейронну мережу або передати значення без перетворень.

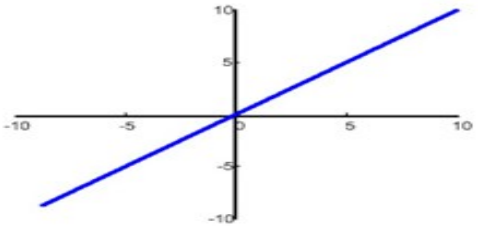


Рисунок 2 – Лінійна функція

$$f(x)=\frac{1}{1-e^{-x}} \tag{4}$$

Сигмоїд (sigmod), формулу 4 та рисунок 3. Це найпоширеніша функція активації, її діапазон значень [0,1]. Саме на ній показано більшість прикладів в мережі, також її іноді називають логістичною функцією. Відповідно, якщо присутні негативні значення (наприклад, акції можуть йти не тільки вгору, але і вниз), то знадобитися функція, яка захоплює і негативні значення.

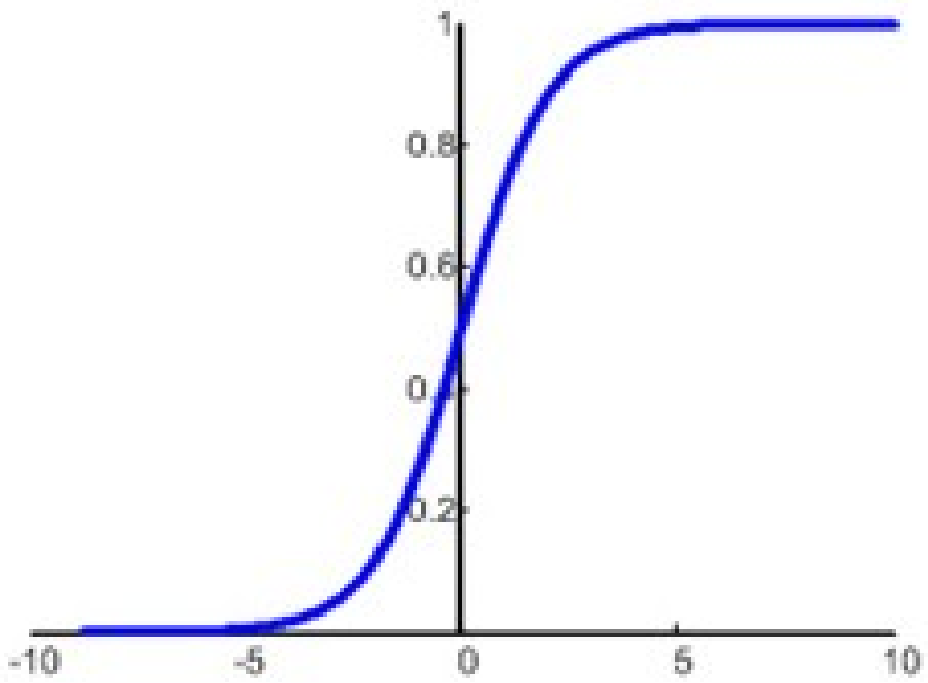


Рисунок 3 – Сигмоїд

$$f(x)=\frac{e^{2x}-1}{e^{2x}+1} \tag{5}$$

Гіперболічний тангенс, формула 5, рисунок 4. Має сенс використовувати гіперболічний тангенс тільки тоді, коли вхідні значення можуть бути і негативними, і позитивними, так як діапазон функції [-1,1]. Викори-

стовувати цю функцію тільки з позитивними значеннями недоцільно, так як це значно погіршить результати нейромережі.

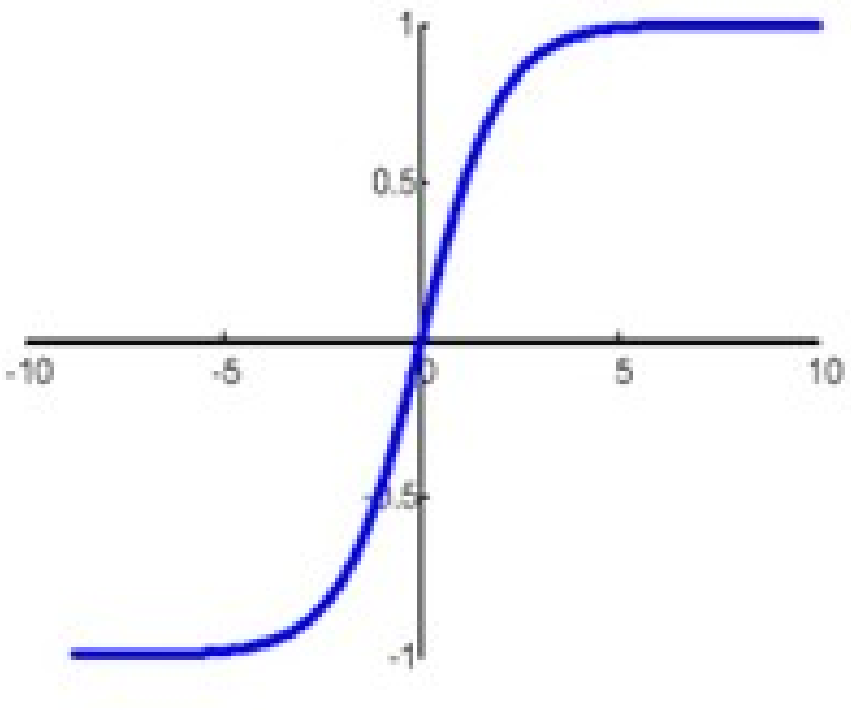


Рисунок 4 – Гіперболічний тангенс

Rectifier linear unit (ReLU) – це урізана лінійна функція, зображена на рисунку 5.

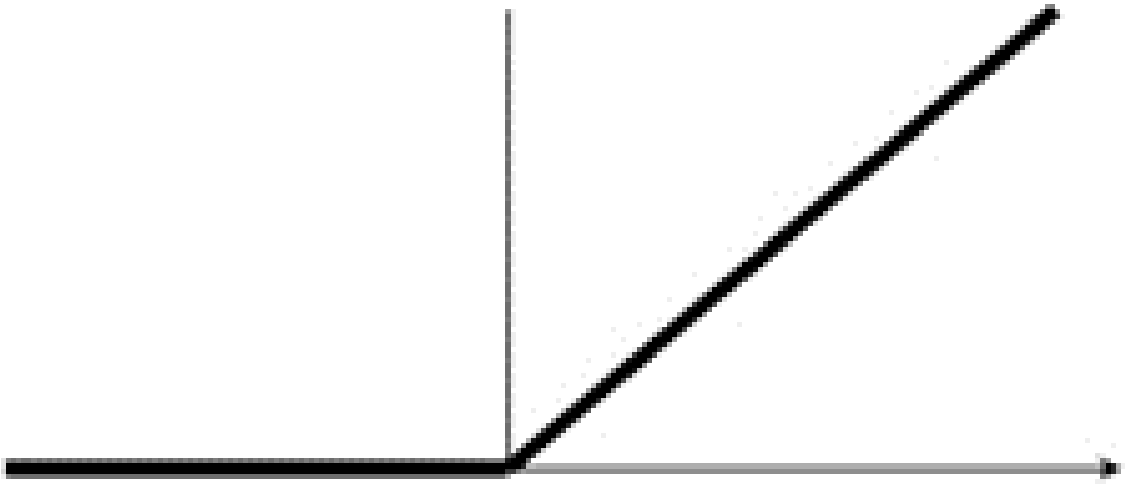

Рисунок 5 – Урізана лінійна функція

Раніше застосовували більш складні нелінійні перетворення, (логістична функція, функція стрибка, кусково-лінійна функція), вони обмежені нулем і одиницею, і ми бачимо, що тут є ділянки лінійності. Тобто вона близько 0 по x поводиться досить лінійно, як звичайна пряма, а далі вона поводиться нелінійно. Але, як виявилося, щоб ефективно навчати подібного роду класифікатори, досить найпростішої нелінійності – просто урізаної прямої (ReLU), коли на додатній ділянці це пряма, а на від'ємній ділянці це завжди 0. Це найпростіша нелінійність, і виявляється, що навіть її вже досить, щоб ефективно навчати класифікацію.

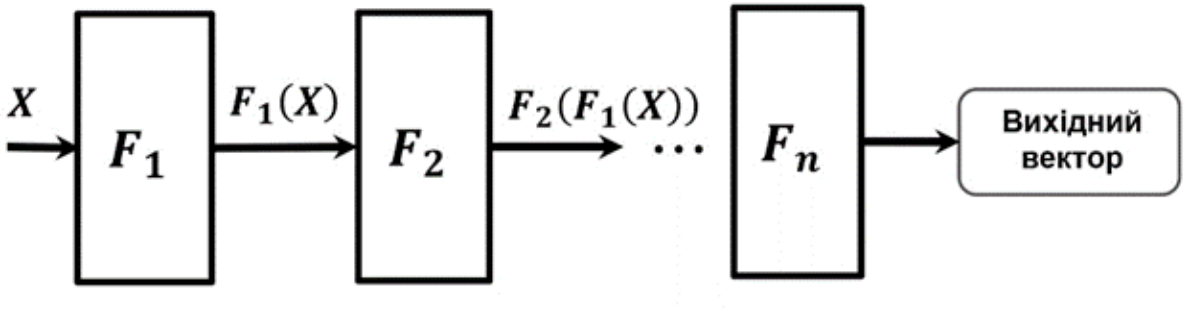


Рисунок 6 – Нейрона мережа, послідовність функціональних перетворень

Нейронна мережа являє собою послідовність перетворень. $F_1$ – це так званий шар нейронної мережі. Шар нейронної мережі – це просто сукупність нейронів, які працюють з одними і тими ж ознаками. Уявімо, що у нас є вихідні ознаки $x_1$, $x_2$, $x_3$, і у нас є три нейрона, кожен з яких пов'язаний з усіма цими ознаками. Але у кожного з нейронів є свої ваги ($w_n$), на яких він зважує ці ознаки, і завдання навчання мережі в підборі таких ваг кожного з нейронів, які б оптимізували цю нашу функцію помилки. І функція $F_1$ – це один шар таких нейронів, після застосування якої отримується якийсь новий простір (вектор, масив, тензор) ознак. Потім до цього простору ознак застосовується ще один такий шар. Там може бути інша кількість нейронів, якась інша нелінійність (функція активації) в якості перетворювальної функції, але це такі ж нейрони, тільки з іншими вагами. Таким чином, послідовно застосовуючи ці перетворення, будується загальна функція F – функція перетворення нейронної мережі, яка складається з послідовного застосування декількох функцій.

Помилка – це процентна величина, що відображає розбіжність між очікуваними та отриманими відповідями. Помилка формується кожну епоху і повинна йти на спад. Якщо цього не відбувається, значить, щось працює не

так. Помилку можна обчислити різними шляхами, але як правило застосовується лише три основних способи: Mean Squared Error (далі MSE, формула 6), Root MSE (формула 7) і Arctan (формула 8). Тут немає якогось обмеження на використання, як у функції активації, і можна вибрати будь-який метод, який буде приносити найкращий результат. Варто лише враховувати, що кожен метод зважує помилки по різному. У Arctan помилка майже завжди буде більше, так як він працює за принципом: чим більша різниця, тим більша помилка. У Root MSE буде найменша помилка, тому, найчастіше, використовують MSE, яка зберігає баланс в обчисленні помилки.

Вище описані формули помилки мають такий вигляд:

$$f(x)=\frac{(I_1-a_1)^2+...+(I_n-a_n)^2}{n} \tag{6}$$

$$f(x)=\sqrt{\frac{(I_1-a_1)^2+...+(I_n-a_n)^2}{n}} \tag{7}$$

$$f(x)=\frac{\arctan(I_1-a_1)^2+...+\arctan(I_n-a_n)^2}{n} \tag{8}$$

Де $I_1$, $I_n$ – це результат розрахунку нейронної мережі, умовні одиниці. $a_1$, $a_n$ – це очікуване значення, умовні одиниці.

Під умовними одиницями може розумітися будь що в залежності від даних, яку аналізує нейрона мережа. Наприклад оцінка валюти через деякий проміжок часу, або клас даних.

### 2.3 Навчання нейронної мережі

Навчання класичної нейронної мережі полягає в розрахунку вагових коефіцієнтів кожного нейрона.

Нехай є набір пар векторів $\{x^\alpha, y^\alpha\}$, $\alpha = 1..p$, так звана навчальна вибірка, що складається з $p$ об'єктів.

Вектор $\{x^\alpha\}$ характеризує систему ознак конкретного об'єкта $\alpha$ навчальної вибірки, зафіксовану вхідними нейронами.

Вектор $\{y^\alpha\}$ характеризує картину збудження нейронів при пред'явленні нейронної мережі конкретного об'єкта $\alpha$ навчальної вибірки.

$x_i^a$ дорівнює 1, якщо в об’єкта $\alpha$ спостерігається ознака i, 0, якщо навпаки.

$y_j^a$ дорівнює 1, якщо при пред’явленні об’єкта $\alpha$ активізується j нейрон, 0, якщо навпаки.

Будемо називати нейронну мережу навченою на даній навчальній вибірці, якщо при подачі на вхід мережі вектора $\{x^\alpha\}$ на виході завжди виходить відповідний вектор $\{y^\alpha\}$, тобто кожному набору ознак відповідають певні класи.

Розенблаттом було запропоновано ітераційний алгоритм навчання з 5ти кроків, який полягає у розрахуванні матриці ваг та послідовно зменшує помилку у вихідних векторах:

1. Початкові значення ваг всіх нейронів закладаються випадковими.
2. Мережі представляється вхідний образ $\{x^\alpha\}$ , та формується вихідний образ.
3. Обчислюється вектор помилки, яка отримана мережею на виході.
4. Вектори вагових коефіцієнтів коригуються таким чином, що величина коригування пропорційна помилці на виході і дорівнює нулю, якщо помилка дорівнює нулю:

   – модифікуються тільки ті компоненти матриці ваг, що відповідають ненульовим значенням входів;

   – знак збільшення ваги відповідає знаку помилки, тобто додатня помилка (значення виходу менше запланованого) приводить до посилення зв'язку;

– навчання кожного нейрона відбувається незалежно від навчання інших нейронів, що відповідає важливому з біологічної точки зору, принципу локальності навчання.

5. Кроки 2-4 повторюються для всіх навчальних векторів. Один цикл послідовного перебору всієї вибірки називається епохою. Навчання завершується після закінчення декількох епох, якщо виконується принаймні одна з умов:

– коли ітерації зійдуться, тобто вектор ваг перестає змінюватися;

– коли повна підсумована по всіх векторах абсолютна помилка стане меншою деякого мінімального значення.

Даний метод навчання був названий Розенблаттом "методом корекції із зворотною передачею сигналу помилки". Мається на увазі передача сигналу помилки від виходу мережі на її вхід, де і визначаються, і використовуються вагові коефіцієнти. Пізніше цей алгоритм назвали "δ- правилом".

Даний алгоритм відноситься до широкого класу алгоритмів навчання з учителем, тому що в ньому вважаються відомими не тільки вхідні вектори, а й значення вихідних векторів, тобто є вчитель, здатний оцінити правильність відповіді учня, причому в якості останнього виступає нейронна мережа.

Розенблаттом доведена "Теорема про збіжність навчання" за δ-правилом. Ця теорема говорить про те, що персептрон здатний навчиться будь-якому навчальному набору, який він здатний уявити.

Одним з найбільш часто використовуваних методів навчання нейромережі - метод зворотного поширення, який використовує алгоритм градієнтного спуску.

Алгоритм градієнтного спуску – це спосіб знаходження локального мінімуму або максимуму функції за допомогою руху вздовж градієнта. Для початку побудується графік, де по осі x будуть значення ваги нейрона (w), а по осі y – помилка, яка відповідає цієї вазі.

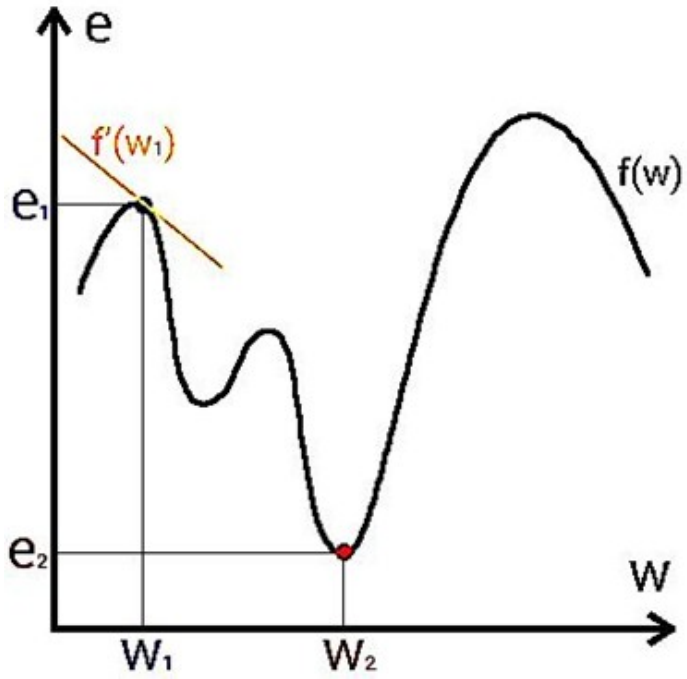


Рисунок 7 – Алгоритм градієнтного спуску

На графіку видно, що графік функції f (w) є залежністю помилки від вибраної ваги. На цьому графіку важливий глобальний мінімум – точка (w2, e2) або, іншими словами, те місце, де графік підходить ближче всього до осі x. Ця точка буде означати, що вибравши вагу w2 отримаємо найменшу помилку e2 і, як наслідок, найкращий результат з усіх можливих. Знайти ж цю точку допоможе метод градієнтного спуску (жовтим на графіку позначений градієнт). Відповідно у кожної ваги в нейромережі буде свій графік і градієнт, і у кожної треба знайти глобальний мінімум.

Градієнт – це вектор який визначає крутизну схилу і вказує його напрямок щодо будь-якої з точок на поверхні або графіку. Щоб знайти градієнт потрібно взяти похідну від графіка в даній точці (як це і показано на графіку). Рухаючись у напрямку цього градієнта ми будемо плавно спускатися вниз, тобто знаходити глобальний мінімум.

Розглянемо як працює метод зворотного поширення.

Після того, як послідовно передається інформація від вхідних нейронів до вихідних і розраховується прогноз, обчислюється помилка і, опираючись на неї, робиться зворотна передача, яка полягає в тому, щоб послідовно змінювати ваги нейронної мережі, починаючи з ваг вихідного нейрона. Значення ваг будуть змінюватися в ту із сторін, яка дасть найкращий результат. Будемо користуватися методом знаходження дельти, так як це найбільш простий і зрозумілий спосіб. Також будемо використовувати стохастичний метод оновлення ваг.

Після розрахунку дельти нейрона потрібно відразу оновити ваги всіх вихідних синапсів цього нейрона. Так як у випадку з вихідним нейроном їх немає, переходимо до нейронів прихованого рівня і робимо те ж саме, за виключенням того, що формула підрахунку дельти тепер (6) і її суть полягає в тому, щоб помножити похідну функції активації від вхідного значення на суму добутків всіх вихідних ваг і дельти нейрона, з якою цей синапс пов'язаний. Але чому формули різні? Справа в тому що вся суть методу зворотного поширення полягає в тому, щоб поширити помилку вихідних нейронів на всі ваги нейромережі. Помилку можна обчислити тільки на вихідному рівні і це вже зроблено. Отже тепер замість помилки далі буде використана дельта, яка буде передаватися від нейрона до нейрона.

Тобто зміна ваги синапсу дорівнює коефіцієнту швидкості навчання, помноженому на градієнт цієї ваги плюс момент помножений на попередню зміну цієї ваги (на 1-ій ітерації дорівнює 0).

Нейромережа можна навчати з вчителем і без (supervised, unsupervised learning).

Навчання з вчителем – це тип тренувань властивий таким проблемам як регресія та класифікація. Для тренування нейромережі треба надати вхідні дані і бажаний результат, тобто мережа, «подивившись» на вхідні дані зрозуміє, що потрібно прагнути до того результату який їй надали.

Навчання без вчителя – цей тип навчання зустрічається не так часто. Тут немає вчителя, тому мережа не отримує бажаний результат, або ж їх кількість дуже мала. В основному такий вид тренувань притаманний нейромережам, у яких завдання полягає в групуванні даних за певними параметрами.

Існує ще метод, як навчання з підкріпленням (reinforcement learning). Такий спосіб застосовується тоді, коли можна, опираючись на результати отримані від нейромережі, дати їй оцінку. Наприклад, ми хочемо навчити нейромережу грати в PAC-MAN, тоді кожен раз коли нейромережа буде набирати багато очок, ми будемо її заохочувати. Іншими словами, ми надаємо нейромережі право знайти будь-який спосіб досягнення мети до того часу, поки він буде давати хороший результат. Таким чином, мережа почне розуміти чого від неї хочуть отримати і намагатиметься знайти найкращий спосіб досягнення цієї мети без постійного надання даних «вчителем».

Також навчання можна проводити трьома способами: стохастичний метод (stochastic), пакетний метод (batch) і міні-пакетний метод (mini-batch).

Стохастичний (його ще іноді називають онлайн) метод працює за наступним принципом – знайшовши $\Delta w$, відразу обнови відповідну вагу.

Пакетний метод же працює по-іншому. Підсумовується $\Delta w$ всіх ваг на поточній ітерації і тільки потім оновлюються всі ваги з використанням цієї суми. Один з найважливіших плюсів такого підходу – це значна економія часу на обчислення, точність ж в такому випадку може дуже постраждати.

Міні-пакетний метод є золотою серединою і намагається поєднати в собі плюси обох методів. Тут принцип такий: у вільному порядку розподіляються ваги по групам і міняються їх ваги на суму $\Delta w$ всіх ваг в тій чи іншій групі.

## 2.4 Згорткові і багатошарові нейронні мережі

### 2.4.1 Багатошаровий персептрон

Часто те, що не вдається зробити відразу, цілком можливо зробити по частинах. Для цього змінюються завдання, які вирішуються шарами нейронної мережі. Виявляється, в першому шарі не слід намагатися на основі первинних ознак, що фіксуються рецепторами, відразу ідентифікувати класи, а потрібно лише сформувати лінійно-роздільну систему вторинних ознак, яку вже на другому шарі зв'язати з класами (рис. 8).

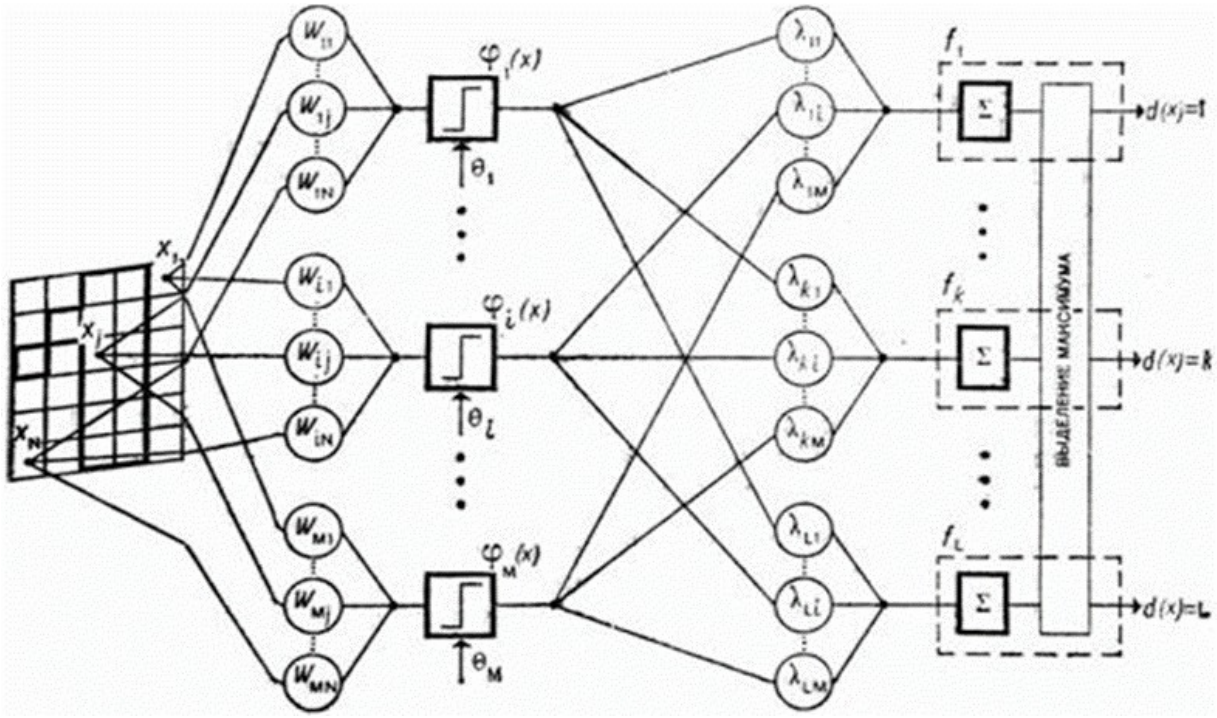


Рисунок 8 – Двох-шаровий персептрон

У багатошаровій мережі вихідні сигнали нейронів попереднього шару відіграють роль вхідних сигналів для нейронів наступного шару, тобто нейрони попереднього шару виступають в якості рецепторів для нейронів наступного шару.

Зв'язки між суміжними шарами нейронів будемо називати безпосередніми, а зв'язки між шарами, розділеними N проміжних шарів, будемо називати зв'язками N-го рівня опосередкованості. Безпосередні зв'язки – це зв'язки 0-го рівня опосередкованості. Проміжні шари нейронів в багатошарових мережах називають прихованими.

Персептрон переводить вхідний образ, що визначає ступені збудження рецепторів, у вихідний образ, який визначається нейронами самого верхнього рівня, яких, як правило, не дуже багато. Стани збудження нейронів на верхньому рівні ієрархії мережі характеризують належність вхідного образу тому чи іншому класу.

Таким чином, багатошаровий персептрон – це навчальна розпізнавальна система, яка реалізує коректовані в процесі навчання лінійні вирішальні правила в просторі вторинних ознак, які, зазвичай, є фіксованими випадково вибраними лінійними граничними функціями від первинних ознак.

При навчанні на вхід персептрона по черзі подаються сигнали з навчальної вибірки, а також вказівки про клас, до якого слід віднести даний сигнал. Навчання персептрону полягає в корекції ваг при кожній помилці розпізнавання, тобто, при кожному випадку розбіжності відповіді, що видається персептроном, і істинного класу. Якщо персептрон помилково відніс сигнал до деякого класу, то ваги функції істинного класу збільшуються, а помилкового зменшуються. У разі правильного рішення всі ваги залишаються незмінними.

Цей надзвичайно простий алгоритм навчання має чудову властивість: якщо існують значення ваг, при яких вибірка може бути розділена безпомилково, то при певних, легко здійснюваних умовах, ці значення будуть знайдені за обмежену кількість ітерацій.

Доведено, що для подання довільного нелінійного функціонального зображення, що задається навчальної вибіркою, достатньо всього двох шарів

нейронів. Однак на практиці, в разі складних функцій, використання більш ніж одного прихованого шару може давати економію загального числа нейронів.

### 2.4.2 Когнітрон і неокогнітрон Фукушіма

В цілому, когнітрон (K.Fukushima, 1975) представляє собою ієрархію шарів, послідовно пов'язаних один з одним, як в персептроні. Однак, при цьому є дві істотні відмінності:

1. Нейрони утворюють не одновимірний ланцюжок, а покривають площину, аналогічно шаруватій будові зорової кори людини.
2. Когнітрон складається з ієрархічно пов'язаних шарів нейронів двох типів – гальмуючих і збудливих.

У когнітроні кожен шар реалізує свій рівень узагальнення інформації:

- вхідні шари чутливі до окремих елементарних структур, наприклад, лініям певної орієнтації або кольору;
- наступні шари реагують вже на більш складні узагальнені образи;
- в шарі найвищого рівня ієрархії активні нейрони визначають результат роботи мережі – розпізнавання певного образу, при цьому результатам розпізнавання відповідають ті нейрони, активність яких виявилася максимальною.

Однак домогтися незалежності (інваріантності) результатів розпізнавання від розмірів і орієнтації зображень вдалося лише в неокогнітроні, який був розроблений Фукушіма в 1980 році і являє собою як би суперпозицію когнітронів, навчених розпізнаванню об'єктів різних типів, розмірів та орієнтації.

### 2.4.3 Згорткові нейронні мережі

Найкращі результати в області розпізнавання осіб показала Convolutional Neural Network або згорткова нейронна мережа (далі - ЗНР), яка є логічним розвитком ідей таких архітектур нейромережі як когнітрона і неокогнітрона. Успіх обумовлений можливістю обліку двовимірної топології зображення, на відміну від багатошарового персептрона.

Згорткові нейронні мережі забезпечують часткову стійкість до змін масштабу, зсувів, поворотів, зміні ракурсу і іншим спотворень. Згорткові нейронні мережі об'єднують три архітектурні ідеї для забезпечення інваріантності до зміни масштабу, повороту, зрушення і просторового спотворення:

- локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів);
- загальні вагові коефіцієнти синапсів (забезпечують детектування деяких рис в будь-якому місці зображення і зменшують загальне число вагових коефіцієнтів);
- ієрархічна організація з просторовими підвибірками.

На даний момент згорткова нейронна мережа і її модифікації вважаються кращими по точності і швидкості алгоритмами знаходження об'єктів на сцені. Починаючи з 2012 року, нейромережі займають перші місця на відомому міжнародному конкурсі з розпізнавання образів ImageNet.

ЗНМ складається з різних видів шарів: згорткові (convolutional) шари, субдискретизуючі (subsampling, підвибірка) шари і шари «звичайної» нейронної мережі – персептрона, відповідно до рисунку 9.

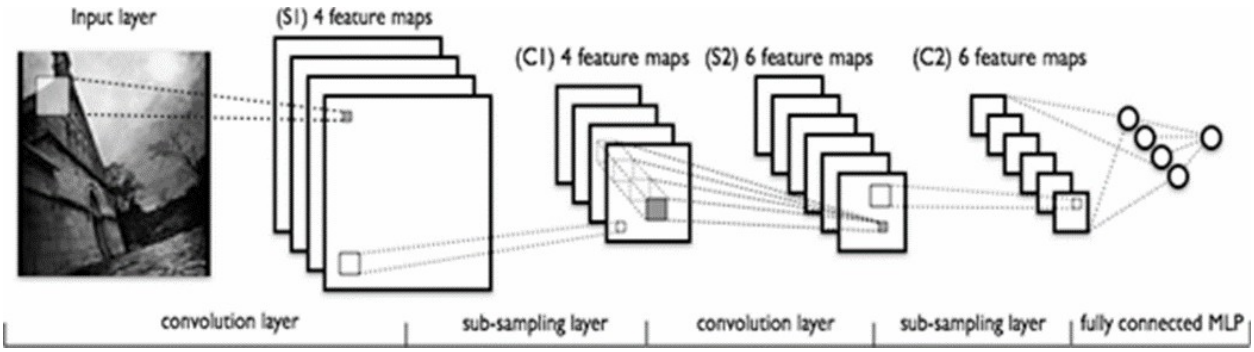


Рисунок 9 – Топологія згорткової нейронної мережі

Перші два типи шарів (convolutional, subsampling), чергуючись між собою, формують вхідний вектор ознак для багатошарового персептрона.

На рисунці нижче продемонстрована візуалізація згортки і підвибірки, зліва зображено вхідне зображення, далі згортка, після цього згортка була опрацьована згідно алгоритму ReLU, далі генеруються під вибірки :

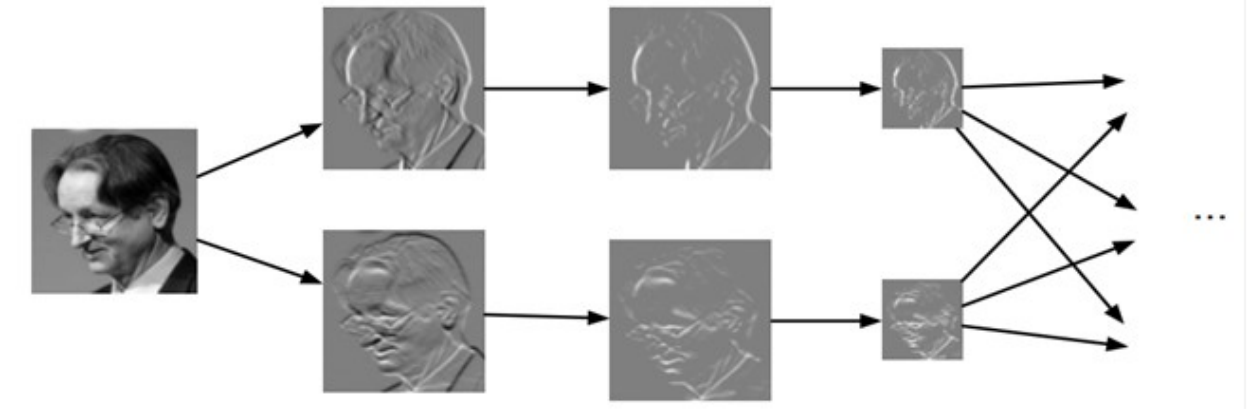

Рисунок 10 – Операція згортки і під вибірки

Згортковий шар являє собою набір карт (інша назва – карти ознак) і це звичайні матриці; у кожної карти є синоптичне ядро (в різних джерелах його називають по-різному: сканувальне ядро або фільтр).

Кількість карт визначається вимогами до задачі, якщо взяти велику кількість карт, то підвищиться якість розпізнавання, але збільшиться обчислювальна складність. Виходячи з аналізу наукових статей, в більшості випадків пропонується брати співвідношення один до двох, тобто кожна карта попереднього шару (наприклад, першого згорткового шару, попереднім є вхідний) пов'язана з двома картами згорткового шару.

Ядро являє собою фільтр або вікно, яке «ковзає» по всій області попередньої карти і знаходить певні ознаки об'єктів. Наприклад, якщо мережу навчали на безлічі осіб, то одне з ядер могло б в процесі навчання видавати найбільший сигнал в області очей, рота, брів або носа, інше ядро могло б виявляти інші ознаки. Розмір ядра зазвичай беруть в межах від 3x3 до 7x7. Якщо розмір ядра маленький, то воно не зможе виділити будь-які ознаки, якщо занадто велике, то збільшується кількість зв'язків між нейронами. Також розмір ядра вибирається таким, щоб розмір карт згорткового шару був парний, це дозволяє не втрачати інформацію при зменшенні розмірності в згортковому шарі.

Ядро являє собою систему поділених ваг або синапсів і це одна з головних особливостей згорткової нейромережі. У звичайній багатошарової мережі дуже багато зв'язків між нейронами, тобто синапсів, що вельми уповільнює процес детектування. У згортковій мережі – навпаки, загальні ваги дозволяють скоротити число зв'язків і знаходити одну й ту саму ознаку по всій області зображення.

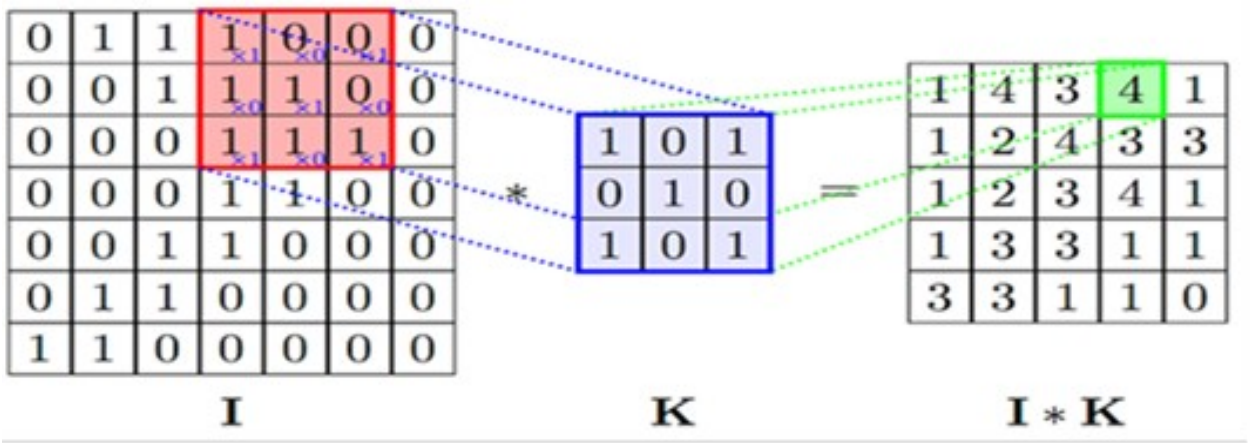


Рисунок 11 – Операція згортки і отримання значення згорткового шару

Операцію згортки можна описати таким чином – вікном розміру ядра g проходимо з заданим кроком (зазвичай 1) все зображення f, на кожному кроці поелементно множимо вміст вікна на ядро g, результат підсумовується і записується в матрицю результату, як на рис. 11.

Де I – це одна з карт попереднього шару, K — ядро, I * K — одна з карт згорткового шару.

Підвибірковий шар також як і згортковий має карти, але їх кількість співпадає з попереднім (згортковим) шаром. Мета шару – зменшення розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки детальне зображення вже не потрібне, і воно ущільнюється до менш детального. До того ж фільтрація вже непотрібних деталей допомагає не перенавчатися.

У процесі сканування ядром підвибіркового шару (фільтром) карти попереднього шару, що скануються ядром, не перетинаються на відміну від згорткового шару. Зазвичай, кожна карта має ядро розміром 2x2, що дозволяє зменшити попередні карти згорткового шару в 2 рази. Вся карта ознак поділяється на осередки 2x2 елементи, з яких вибираються максимальні за значенням.

Зазвичай в підвибірковому шарі застосовується функція активації ReLU. MaxPooling – операція підвибірки з вибором максимального.

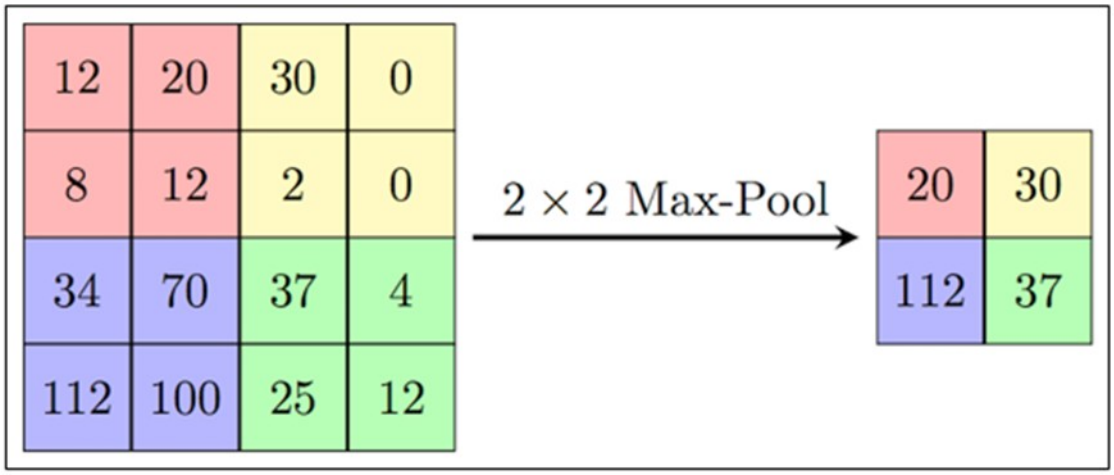


Рисунок 12 – Формування нової карти підвибіркового шару на основі попередньої карти згорткового шару. Операція підвибірки (Max Pooling)

# 3 АНАЛІЗ СТАТИСТИЧНИХ ТА МАТЕМАТИЧНИХ МЕТОДІВ

## 3.1 Використання статистики у контексті нейронних мереж

Статисти́ка −наука, що вивчає методи кількісного охоплення і дослідження масових, зокрема суспільних, явищ і процесів.

Історично, у науку термін «статистика» ввів німецький учений Готфрід Ахенвалль в 1746 році, запропонувавши замінити назву курсу «Державознавство», що викладалося в університетах Німеччини, на «Статистику», поклавши тим самим початок розвитку статистики як науки й навчальної дисципліни. Незважаючи на це, статистичний облік вівся набагато раніше: проводилися переписи населення в Стародавньому Китаю, здійснювалося порівняння військового потенціалу держав, вівся облік майна громадян в Стародавньому Римі тощо.

Статистика розробляє спеціальну методологію дослідження й обробки матеріалів: масові статистичні спостереження, метод угруповань, середніх величин, індексів, балансовий метод, метод графічних зображень та інші методи аналізу статистичних даних.

Збирання інформації в різних областях людської діяльності сягає найдавніших часів. Вона мала спершу наскрізь практичний характер; з XIX ст. статистика поступово здобуває солідну наукову основу, коли почалося впорядкування і вдосконалення статистичних методів. ІЗ них розвинулися дві основні: описова (дескриптивна) – збирання інформації, перевірка її якості, її інтерпретація, зображення статистичного матеріалу; та індуктивна — застосування теорії ймовірності, закону великих чисел. Статистика поділяється за своїм змістом на демографічну, економічну, фінансову, соціальну, санітарну, судову, біологічну, технічну тощо; математична статистика вивчає математичні методи систематизації, обробки й використання статистичних даних для наукових і практичних висновків.

Мета кожного наукового дослідження – виявлення закономірностей явищ, які спостерігають, та використання цих закономірностей у повсякденній практичній діяльності. Для встановлення цих закономірностей проводять спеціальні досліди та спостерігають одиничні явища. Далі роблять узагальнений висновок у вигляді закону.

У тих випадках, коли явище знаходиться під дією багатьох факторів і неможливо виявити вплив усіх цих факторів, застосовують інший метод вивчення – статистичний, тобто систематизація та обробка статистичних даних однорідних дослідів.

Основні задачі, які розв’язує математична статистика:

– вказати способи збору та групування (якщо даних дуже багато) статистичних відомостей;

– визначити закон розподілу випадкової величини або системи випадкових величин за статистичними даними;

– визначити невідомі параметри розподілу;

– перевірити правдоподібність припущень про закон розподілу випадкової величини, про форму зв’язку між випадковими величинами або про значення параметра, який оцінюють.

В структурі математичної статистики традиційно виділяють два основні розділи: описова статистика і статистичні висновки.

Описова статистика використовується для:

– узагальнення показників однієї змінної (статистика випадкової вибірки);

– виявлення взаємозв'язків між двома і більше змінними (кореляційно-регресійний аналіз).

Описова статистика дає можливість отримати нову інформацію, швидше зрозуміти і всебічно оцінити її, тобто, виконує наукову функцію опису об'єктів дослідження, чим і виправдовує свою назву. Методи описової статистики покликані перетворити сукупність окремих емпіричних даних на си-

стему наочних для сприйняття форм і чисел: розподіли частот; показники тенденцій, варіативності, зв'язку. Цими методами розраховуються статистики випадкової вибірки, які служать підставою для здійснення статистичних висновків.

Статистичні висновки надають можливість:

- оцінити точність, надійність і ефективність вибіркових статистик, виявити похибки, які виникають у процесі статистичних досліджень (статистичне оцінювання);
- узагальнити параметри генеральної сукупності, отримані на підставі вибіркових статистик (перевірка статистичних гіпотез).

Генеральна сукупність − це повна сукупність об'єктів дослідження; вибірка − це та її частина, яка сформована певним науково обґрунтованим способом.

Генеральні сукупності можуть набувати значних розмірів, бути скінченими і нескінченими. На практиці, як правило, мають справу зі скінченими сукупностями. І якщо відношення обсягу генеральної сукупності до обсягу вибірки складає більш, ніж 100, то, методи оцінювання для скінчених і нескінчених сукупностей дають по суті однакові результати. Генеральною сукупністю можна називати і повну сукупність значень якоїсь ознаки. Належність вибірки до генеральної сукупності є головною підставою для оцінки характеристик генеральної сукупності за характеристиками вибірки.

Основна ідея математичної статистики базується на переконанні про те, що повне вивчення всіх об'єктів генеральної сукупності в більшості наукових завдань або практично неможливе, або економічно недоцільно, оскільки вимагає багато часу і значних матеріальних витрат.

## 3.2 Чисельні методи у контексті нейроних мереж. Кореляція.

Кореляція, кореляційний залежність - взаємозалежність двох або кількох випадкових величин. Суть її полягає в тому, що при зміні значення однієї змінної відбувається закономірна зміна (зменшення або збільшення) іншої (-их) змінної (-их).

При розрахунку кореляцій намагаються визначити, чи існує статистично достовірний зв'язок між двома або кількома змінними в одній або декількох вибірках. Наприклад, взаємозв'язок між зростанням і вагою дітей, взаємозв'язок між успішністю і результатами виконання тесту IQ, між стажем роботи і продуктивністю праці.

Важливо розуміти, що кореляційна залежність відображає тільки взаємозв'язок між змінними і не говорить про причинно-наслідкові зв'язки. Наприклад, якби досліджуваної вибірці між зростом і вагою людини існувала кореляційний залежність то, це не означало б, що вага є причиною зростання людини, інакше скидаючи зайві кілограми зростання людини також зменшувався. Кореляційний зв'язок лише говорить про взаємозв'язку даних параметрів, причому в даній конкретній вибірці, в іншій вибірці ми можемо не спостерігати отримані кореляції.

Показник кореляції. Коефіцієнт кореляції (r) характеризує величину відображає ступінь взаємозв'язку двох змінних між собою. Він може варіювати в межах від -1 (негативна кореляція) до +1 (позитивна кореляція). Якщо коефіцієнт кореляції дорівнює 0 то, це говорить про відсутність кореляційних зв'язків між змінними. Причому якщо коефіцієнт кореляції ближче до 1 (або -1) то кажуть про сильної кореляції, а якщо ближче до 0, то про слабку.

При позитивної кореляції збільшення (або зменшення) значень однієї змінної веде до закономірного збільшення (або зменшення) іншої змінної тобто взаємозв'язку типу збільшення-збільшення (зменшення-зменшення).

При негативній кореляції збільшення (або зменшення) значень однієї змінної веде до закономірного зменшення (або збільшення) іншої змінної тобто взаємозв'язку типу збільшення-зменшення (зменшення-збільшення).

### 3.3 Стандартне відхилення (девіація) у контексті нейронних мереж

У статистиці стандартне відхилення – це міра величини варіації або дисперсії набору значень. Низьке стандартне відхилення вказує на те, що значення мають тенденцію бути близькими до середнього (також називається очікуваним значенням) набору, тоді як високе стандартне відхилення вказує на те, що значення розповсюджені на більш широкий діапазон.

Стандартне відхилення може бути скороченим SD і найчастіше представлене в математичних текстах та рівняннях нижньою регістровою грецькою буквою сигма σ, для стандартного відхилення сукупності або латинською буквою s, для вибірки стандартного відхилення.

Стандартне відхилення випадкової величини, вибірки, статистичної сукупності, набору даних або розподілу ймовірностей є квадратним коренем її дисперсії. Це алгебраїчно простіше, хоча на практиці воно менш надійне, ніж середнє абсолютне відхилення. Корисною властивістю стандартного відхилення є те, що на відміну від дисперсії, воно виражається в тій же одиниці, що і дані.

Стандартне відхилення сукупності чи вибірки та стандартна похибка статистики (наприклад, середнього значення вибірки) досить різні, але пов’язані між собою. Стандартна помилка вибірки - це стандартне відхилення набору середніх значень, яке було б знайдено шляхом вилучення нескінченної кількості повторюваних вибірок із сукупності та обчислення середнього значення для кожної вибірки. Стандартна похибка середнього значення дорівнює стандартному відхиленню сукупності, поділеному на квадратний корінь розміру вибірки, і оцінюється за допомогою стандартного

відхилення вибірки, поділеного на квадратний корінь розміру вибірки. Наприклад, стандартна помилка опитування (що повідомляється як межа похибки опитування) - це очікуване стандартне відхилення розрахункового середнього значення, якщо одне і те ж опитування проводитиметься кілька разів. Таким чином, стандартна помилка оцінює стандартне відхилення оцінки, яка сама вимірює, наскільки оцінка залежить від конкретної вибірки, взятої з сукупності.

У науці прийнято повідомляти як про стандартне відхилення даних (як сумарну статистику), так і про стандартну похибку оцінки (як міру потенційної похибки у результатах). За умовою, лише ефекти, що відрізняються від більш ніж двох стандартних помилок від нульового очікування, вважаються "статистично значущими", що є гарантією від хибних висновків, які насправді обумовлені помилкою випадкової вибірки.

У статистиці коефіцієнт кореляції Пірсона [10] також відомий як r Пірсона, коефіцієнт кореляції продукту-моменту Пірсона, двомірний кореляційний зв'язок, або просто, просто як коефіцієнт кореляції – це міра лінійної кореляції між двома наборами даних. Це співвідношення між коваріантністю двох змінних та добутком їх стандартних відхилень; таким чином, це, по суті, нормалізоване вимірювання коваріації, так що результат завжди має значення між − 1 і 1. Як і у випадку самої коваріації, міра може відображати лише лінійну кореляцію змінних та ігнорує багато інших типів зв’язків або кореляцій . Як простий приклад, можна було б очікувати, що вік та зріст вибірки підлітків із середньої школи мають коефіцієнт кореляції Пірсона значно більший за 0, але менший за 1 (оскільки 1 буде представляти нереально досконалу кореляцію).

У статистиці коефіцієнт рангової кореляції Спірмена або ρ Спірмена, названий на честь Чарльза Спірмана і часто позначається грецькою буквою ρ (rho) або як $r_s$, є непараметрична міра рангової кореляції (статистична залеж-

ність між ранжуванням двох змінних). Він оцінює, наскільки добре відносини між двома змінними можна описати за допомогою монотонної функції.

Кореляція Спірмена між двома змінними дорівнює кореляції Пірсона між значеннями рангу цих двох змінних; тоді як кореляція Пірсона оцінює лінійні відносини, кореляція Спірмена оцінює монотонні відносини (лінійні чи ні). Якщо немає повторюваних значень даних, ідеальна кореляція Спірмена +1 або −1 має місце, коли кожна зі змінних є ідеальною монотонною функцією іншої.

Інтуїтивно, кореляція Спірмена між двома змінними буде високою, коли спостереження мають схожий (або ідентичний для кореляції 1) ранг (тобто позначка відносного положення спостережень у межах змінної: 1 -й, 2 -й, 3 -й тощо) між двома змінні, і низькі, коли спостереження мають несхожий (або повністю протилежний кореляції −1) ранг між двома змінними.

### 3.4 Використання чисельних методів у контексті обробки та підготовки даних

Дуже часто дані можуть містити аномалії, шуми або відсутні комірки у вибірці. Непідготовлений набір даних може вплинути на результати дослідження починаючи від доказу теорії чи її дослідження закінчуючи неправильному навчанню нейронної мережі або іншої одиниці машинного навчання.

Для того щоб вирішити цю проблему необхідно до кожної проблеми застосувати необхідний інструмент. Наприклад, для відсутніх даних можна застосувати відновлення. Для відновлення використовують, зазвичай, середнє значення з іншого локального інтервалу чи подібного історичного, але це може знизити точність досліджень. Більш правильний метод – це застосування інтерполяції.

Інтерполяція, інтерполювання [11] – в обчислювальної математики знаходження невідомих проміжних значень деякої функції, за наявним дис-

кретному набору її відомих значень, певним способом. Термін «інтерполяція» запустив у вжиток Джон Валліс в своєму трактаті «Арифметика нескінченних» (1 656).

У функціональному аналізі інтерполяція лінійних операторів є розділ, який би розглядав банахови простору як елементи деякої категорії.

Багатьом з тих, хто стикається з науковими та інженерними розрахунками, часто доводиться оперувати наборами значень, отриманих дослідним шляхом або методом випадкової вибірки. Як правило, на підставі цих наборів потрібно побудувати функцію, на яку могли б з високою точністю потрапляти інші одержувані значення. Таке завдання називається апроксимацією. Інтерполяцією називають такий різновид апроксимації, при якій крива побудованої функції проходить точно через наявні точки даних.

Існує також близька до інтерполяції задача, яка полягає в апроксимації будь-якої складної функції іншої, більш простою функцією. Якщо деяка функція занадто складна для продуктивних обчислень, можна спробувати обчислити її значення в декількох точках, а по ним побудувати, тобто інтерполювати, більш просту функцію. Зрозуміло, використання спрощеної функції не дозволяє отримати такі ж точні результати, які давала б початкова функція. Але в деяких класах завдань досягнутий виграш в простоті і швидкості обчислень може переважити отримувану похибка в результатах.

Слід також згадати і зовсім інший різновид математичної інтерполяції, відому під назвою «інтерполяція операторів». До класичних робіт по інтерполяції операторів відносяться теорема Рісса - Торіна і теорема Марцинкевича, що є основою для безлічі інших робіт.

Існують такі види інтерполяції:

- Лінійна інтерполяція
- Інтерполяційна формула Ньютона
- Метод кінцевих різностей
- Многочлен Лагранжа

- Схема Ейткена
- Сплайн-функція
- Кубічний сплайн
- Поліном Лагранжа
- Зворотне інтерполювання по формулі Ньютона
- Зворотне інтерполювання по формулі Гаусса
- Білінійна інтерполяція
- Бікубічна інтерполяція
- Раціональна інтерполяція
- Тригонометрична інтерполяція

Шум [12] – це безладні коливання різної фізичної природи, що відрізняються складністю тимчасової і спектральної структури. Спочатку слово шум належало виключно до звукових коливань, проте в сучасній науці воно було поширене і на інші види коливань (радіо-, електрика).

# 4 ВИРІШЕННЯ ПРАКТИЧНИХ ЗАВДАНЬ

## 4.1 Інструментарій побудови нейронних мереж і збору статистики

Tensorflow (далі – TF) – досить молодий фреймворк (API високого рівня з підтримкою мов програмування: python, java, c ++, javascript, go, swift) для глибокого машинного навчання, що розробляється в Google Brain. Довгий час фреймворк розроблявся в закритому режимі під назвою DistBelief, але після глобального рефакторінга 9 листопада 2015 року був випущений в open source. За рік з невеликим TF доріс до версії 1.0, знайшов інтеграцію з keras, став значно швидшим і отримав підтримку мобільних платформ.

Робота с TF будується навколо побудови і виконання графа обчислень. Граф обчислень – це конструкція, яка описує те, яким чином будуть проводитися обчислення. У класичному імперативному програмуванні ми пишемо код, який виконується послідовно. В TF звичний імперативний підхід до програмування необхідний тільки для якихось допоміжних цілей. Основа TF – це створення структури, яка задає порядок обчислень. Програми структуруються на дві частини – складання графа обчислень і виконання обчислень в створених структурах.

В TF граф складається з плейсхолдерів, змінних та операцій. З цих елементів можна зібрати граф, в якому будуть обчислюватися тензори. Тензори – багатовимірні масиви, вони служать «паливом» для графа. Тензором може бути як окреме число, вектор ознак розв'язуваної задачі або зображення, так і цілий Батч описів об'єктів або масив із зображень. Замість одного об'єкта ми можемо передати в граф масив об'єктів і для нього буде вирахувано масив відповідей. Робота TF з тензорами схожа на те, як обробляє масиви numpy, у функціях якого можна вказати вісь масиву, щодо якої буде виконуватися обчислення.

На більш високих рівнях можна просто оголошувати тип шару, його характеристики і функції, а TF сам все зробить. У TF є три основні режими: тренування (train, fit), перевірки (eval) і передбачення (predict) для одиничного входу. При цьому є можливість збирати різного роду статистику, а потім її обробляти. В TF присутній модуль Tensorboard, який дозволяє в браузері переглядати, як виглядає граф обчислень, будувати графіки різних змінних і таким чином відслідковувати роботу нейронної мережі.

TF дозволяє будувати, навчати і тестувати нейронні мережі великою кількістю способів. На самому нижньому простому рівні можна для кожного шару задавати масиви ваг і зміщень для нейронів. Потім за допомогою вбудованих математичних функцій всіх їх перемножувати і підсумовувати. Після підсумовування отримати за допомогою декількох функцій активації пророкування класу. Також є вбудований набір алгоритмів по виділенню помилок і точності, а також різних алгоритмів навчання.

Ще однією чудовою властивістю TF є можливість запису/ завантаження стану моделі. Тому можна поетапно тренувати модель, а також тренувати модель на потужному комп'ютері, а тестувати і використовувати вже натреновані моделі на мобільному телефоні чи ноутбуці. Таким чином в роботі для навчання моделей використовувався комп'ютер з графічною картою NVIDIA GeForce 1060 з підтримкою CUDA. Це значно прискорювало процес, так як обчислення виконувались графічною картою з великою кількістю процесорів. Те, що на відеокарті розраховувалось за 20-30 хвилин, на ноутбуці (вбудована відеокарта) тренувалося 1,5-2 доби.

### 4.2 Набори символів, які досліджувалися

Для тренування і тестування нейронної мережі необхідні набори вхідних даних. Чим більший набір для тренування, тим якісніше буде навчена нейромережа. Для оцінки якості навчання нейромережі використовується

тестовий набір, при цьому основна вимога – це незалежність наборів для тесту і навчання, тобто в них повинні бути різні дані. Для тестування роботи мережі зазвичай використовують базу MNIST.

База даних MNIST (скорочення від «Modified National Institute of Standards and Technology») – об'ємна база даних зразків рукописного написання цифр. База даних є стандартом, запропонованим Національним інститутом стандартів і технологій США з метою калібрації і зіставлення методів розпізнавання зображень за допомогою машинного навчання в першу чергу на основі нейронних мереж. Дані складаються із заздалегідь підготовлених прикладів зображень, на основі яких проводиться навчання та тестування систем [8] [9]. База даних була створена після переробки оригінального набору чорно-білих зразків розміром 20x20 пікселів NIST. Творці бази даних NIST, в свою чергу, використовували набір зразків з Бюро перепису населення США, до якого були додані ще тестові зразки, написані студентами американських університетів. Зразки з набору NIST були нормалізовані, пройшли згладжування і приведені до сірого напівтонування розміром 28x28 пікселів.

База даних MNIST містить 55000 (60000) зображень для навчання і 10000 зображень для тестування. Половина зразків для навчання і тестування були взяті з набору NIST для навчання, а інша половина – з набору NIST для тестування. Крім того в системі API Tensorflow є приклади з використання цієї бази рукописних цифр.

Але цього замало, так як одним із завдань даної роботи є розпізнавання рукописних букв. З цією метою була використана база рукописних цифр і букв EMNIST [9,10] (Extended MNIST). EMNIST база, яка була форматована і підготовлена, як і MNIST.

База EMNIST містить кілька видів наборів, які відрізняються, кількістю класів:

- EMNIST ByClass: 814,255 символів, 62 незбалансованих класи

- EMNIST ByMerge: 814,255 символів, 62 незбалансованих класи
- EMNIST Balanced: 131,600 символів. 47 збалансованих класів
- EMNIST Letters: 145,600 символів. 26 збалансованих класів
- EMNIST Digits: 280,000 символів. 10 збалансованих класів
- EMNIST MNIST: 70,000 символів. 10 збалансованих класів

Таблиця 4.1 – Кількість символів в різних наборах NIST Database 19, яка використовувалася для створення EMNIST.

| | Тип | Число класів | Тренувальні | Тестові | Всього |
|---|---|---|---|---|---|
| By Class | Цифри | 10 | 344,307 | 58,646 | 402,953 |
| | Прописні символи | 26 | 208,363 | 11,941 | 220,304 |
| | Рядкові символи | 26 | 178,998 | 12,000 | 190,998 |
| | Разом: | 62 | 731,668 | 82,587 | 814,255 |
| By Merge | Цифри | 10 | 344,307 | 58,646 | 402,953 |
| | Символи | 37 | 387,361 | 23,941 | 411,302 |
| | В сумі: | 47 | 731,668 | 82,587 | 814,255 |

By Class (по класах): це найкорисніша організація даних з точки зору класифікації, оскільки вона містить сегментовані цифри і символи, впорядковані за класами. Таких класів 62, що містять символи [0-9], [a-z] і [A-Z].

By Merge (сполучена): ця ієрархія даних звертається до проблеми в класифікації рукописних цифр, які мають схожість з великими та малими бу-

квами. Це варіант, коли в наборі даних об'єднуються певні класи, створюючи 47 класів. Об'єднані класи, як це було запропоновано NIST, призначені для букв C, I, J, K, L, M, O, P, S, U, V, W, X, Y і Z.

Згідно рекомендаціями до навчання нейронних мереж необхідно з набору даних виділити частину тренувальних та тестових даних. У даному випадку було використано таку комбінацію: 5-7% тестових даних та 93-95% тренувальних даних.

Таблиця 4.2 – Структура і організація в наборі EMNIST

| Назва | Класи | Тренувальних | Тестових | **Всього** |
|---|---|---|---|---|
| Complete | 62 | 697,932 | 116,323 | 814,255 |
| Merge | 47 | 697,932 | 116,323 | 814,255 |
| Balanced | 47 | 112,800 | 18,800 | 131,600 |
| Digits | 10 | 240,000 | 40,000 | 280,000 |
| Letters | 37 | 88,800 | 14,800 | 103,600 |
| MNIST | 10 | 60,000 | 10,000 | 70,000 |

На рис. 13 проілюстровані приблизні гістограми і алгоритм «поєднання» окремих класів. Кожен набір даних містить рукописні цифри, рукописні літери або комбінацію обох.

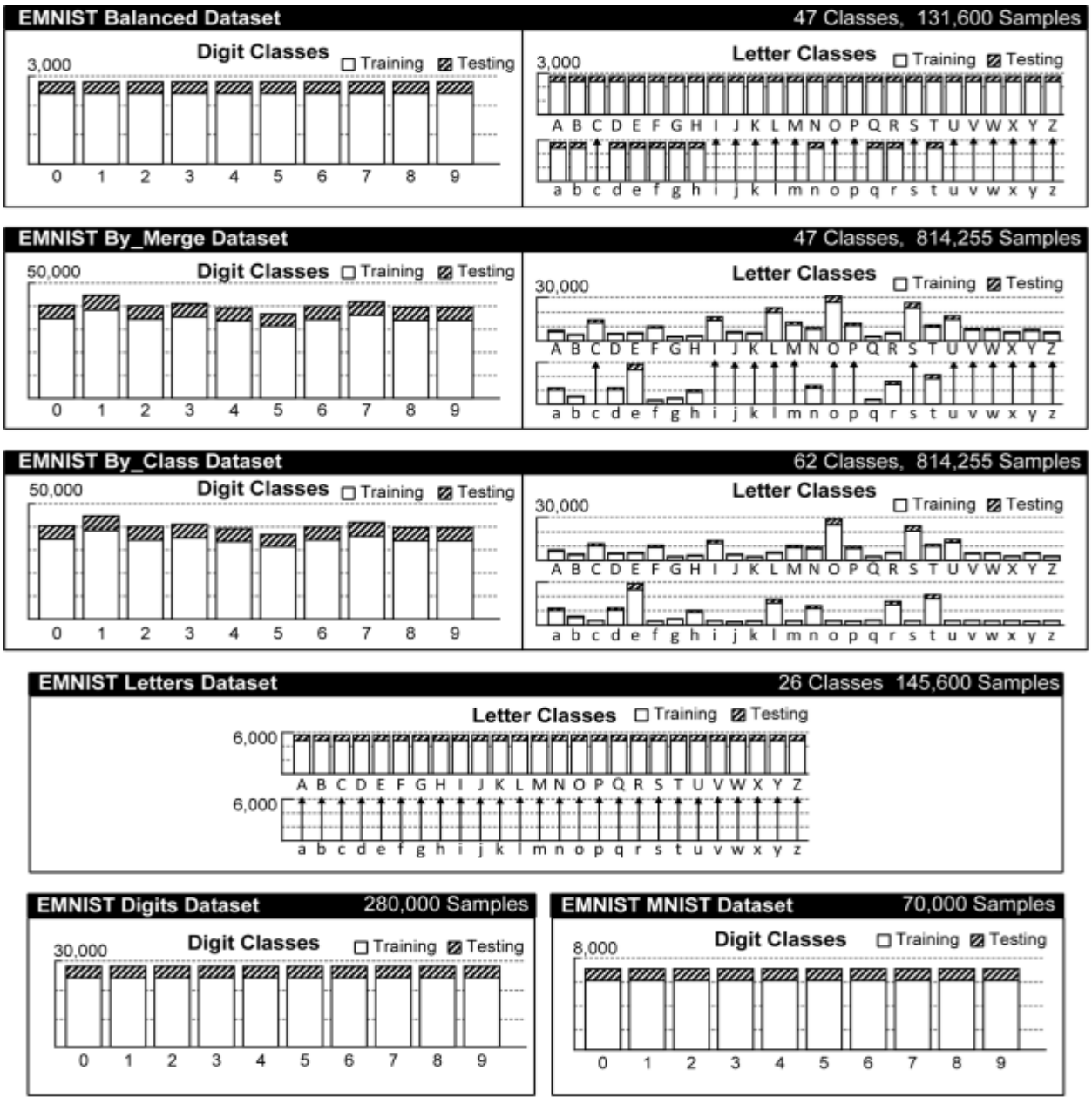


Рисунок 13 – Візуальне розбиття наборів даних EMNIST

### 4.3 Види нейронних мереж, які досліджувалися

В літературі наводяться дані про структуру і розмір різних нейронних мереж і точності розпізнавання. Структури, які використовуються у поточному проекті наведені в таблиці 4.3.

Таблиця 4.3 – Приклади результатів машинного навчання в різних системах класифікації зображень

| Тип | Структура | Спотворення | Попередня обробка | Помилка (%) |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| Лінійний класифікатор | однорівневий персептрон | Ні | Ні | 12 |
| Метод k найближчих сусідів | K-NN з нелінійної деформацією (P2DHMDM) | Немає | Shiftable edges | 0.52 |
| Gradient boosting | Обробка залишків на базі ознак Хаара | Немає | Ознаки Хаара | 0.87 |
| Нелінійний класифікатор | 40 PCA + квадратичний класифікатор | Ні | Ні | 3.3 |
| Метод опорних векторів | Віртуальна система опорних векторів, deg-9 poly, 2-pixel jittered | Ні | Вирівнювання | 0.56 |
| Нейронна мережа | 2-рівнева мережа | Ні | Ні | 1.6 |
| Нейронна мережа | 2-рівнева мережа 784-800-10 | Пружні деформації | Немає | 0.7 |

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Глибока нейронна мережа | 6-рівнева мережа 784-2500-2000-1500-1000-500-10 | Пружні деформації | Ні | 0.35 |
| Згорткова нейронна мережа | 6-рівнева мережу 784-40-80-500-1000-2000-10 | Ні | Розширення даних для навчання | 0.31 |
| Згорткова нейронна мережа | Ансамбль з 35 CNN-мереж, 1-20-P-40-P-150-10 | Пружні деформації | З нормалізацією | 0.23 |
| Згорткова нейронна мережа | Ансамбль з 5 CNN-мереж, 6-рівнів 784-50-100-500-1000-10-10 | Ні | Розширення даних для навчання | 0.21 |
| Випадкове мультимодальне глибоке навчання (RMDL) | 30 моделей випадкового глибокого «підскакування» (RDL) (10 CNN, 10 RNN і 10 DNN) | Ні | Ні | 0.18 |

На рис. 14 показана залежність точності розпізнавання для нейронних мереж в залежності від складності нейромережі для різних наборів даних.

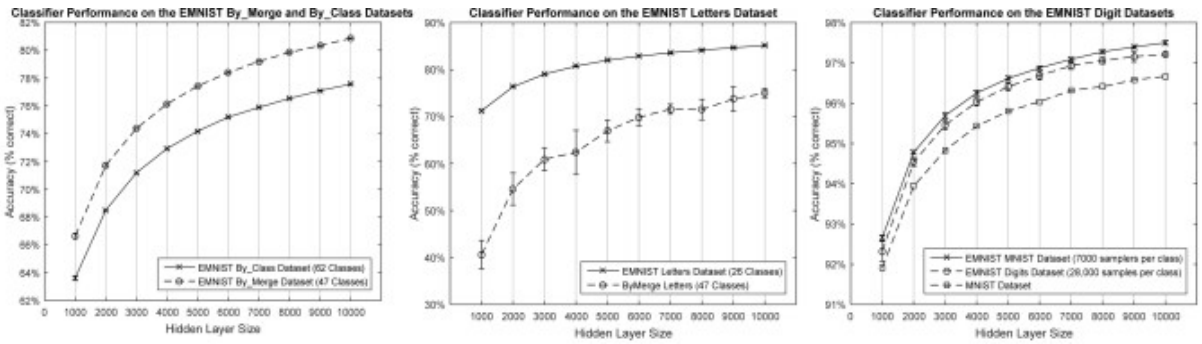


Рисунок 14 – Залежність точності розпізнавання в залежності від складності нейромережі для різних наборів даних

Як бачимо, що чим складніша нейромережа, тим вища точність розпізнання. Однак при значному зростанні складності мережі, часто «битва» йде за десяті частини відсотка.

Першою досліджуваною моделлю стала нейронна мережа з прихованим шаром (персептрон). Властивості шару і функції обробки виконані в явному вигляді, для демонстрації можливостей контролю нейронної мережі на найнижчому рівні. Роздруківка модуля міститься в Додатку 1.

Даний приклад являє інтерес тільки в якості вивчення, тобто як працює нейронна мережа всередині, як збирається статистика по вагам, верствам, зрушенням.

На рис 15 зображена гістограма, активації вихідного шару.

З цієї гістограми видно, що зі збільшенням числа кроків, тренованість мережі зростає і ймовірність передбачення правильної відповіді зростає.

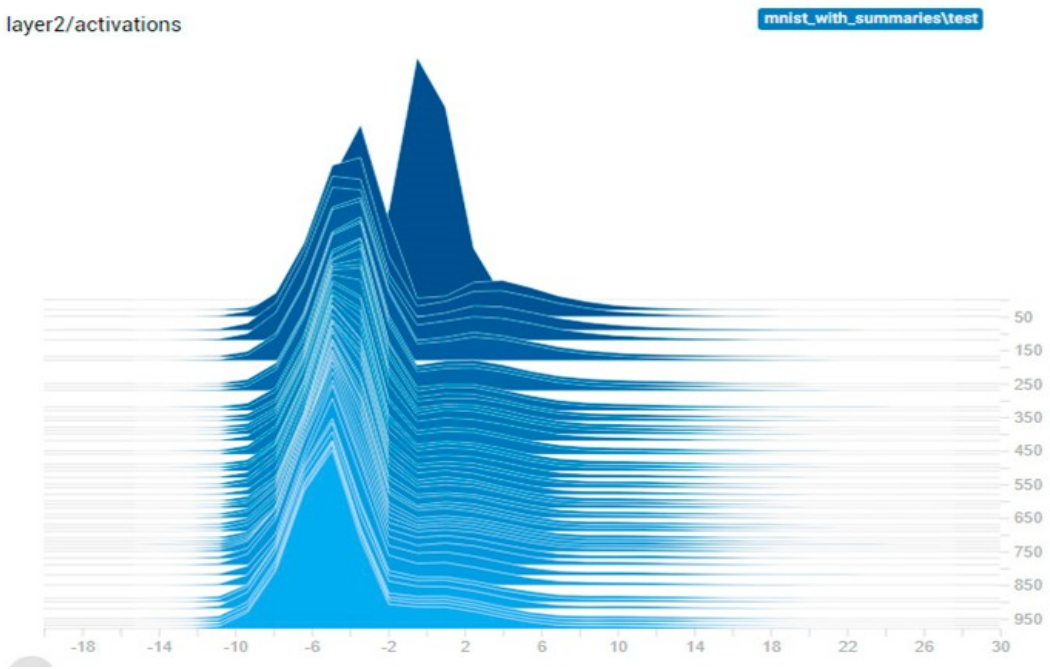


Рисунок 15 – Гістограма активації вихідного шару

На рис 15 і 16 наведено графік зміни точності і втрат (розріджена крос-ентропію) в процесі тренування і тестування нейронної мережі.

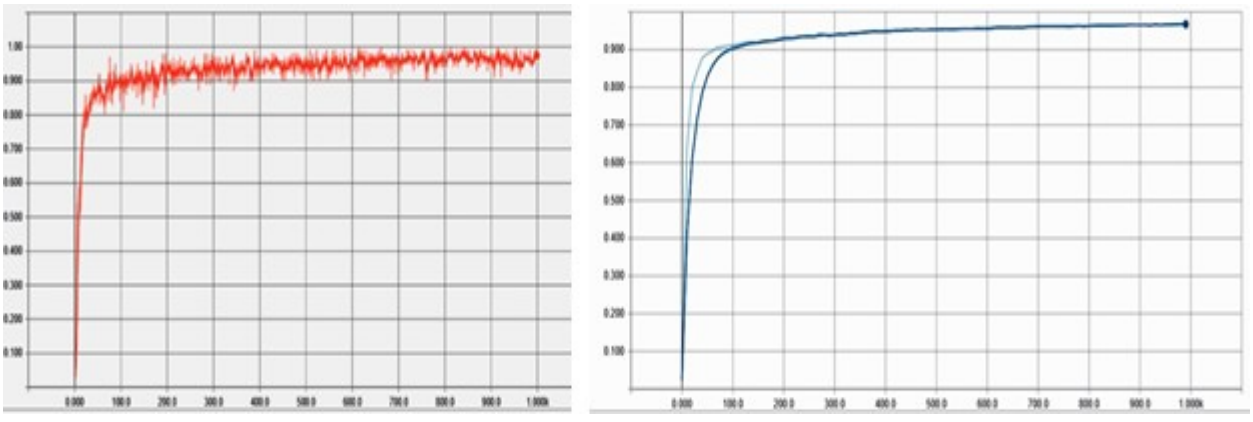

Рисунок 15 – Графік зміни точності (accuracy) в залежності від кількості кроків: при тренуванні (зліва), при тестуванні (праворуч)

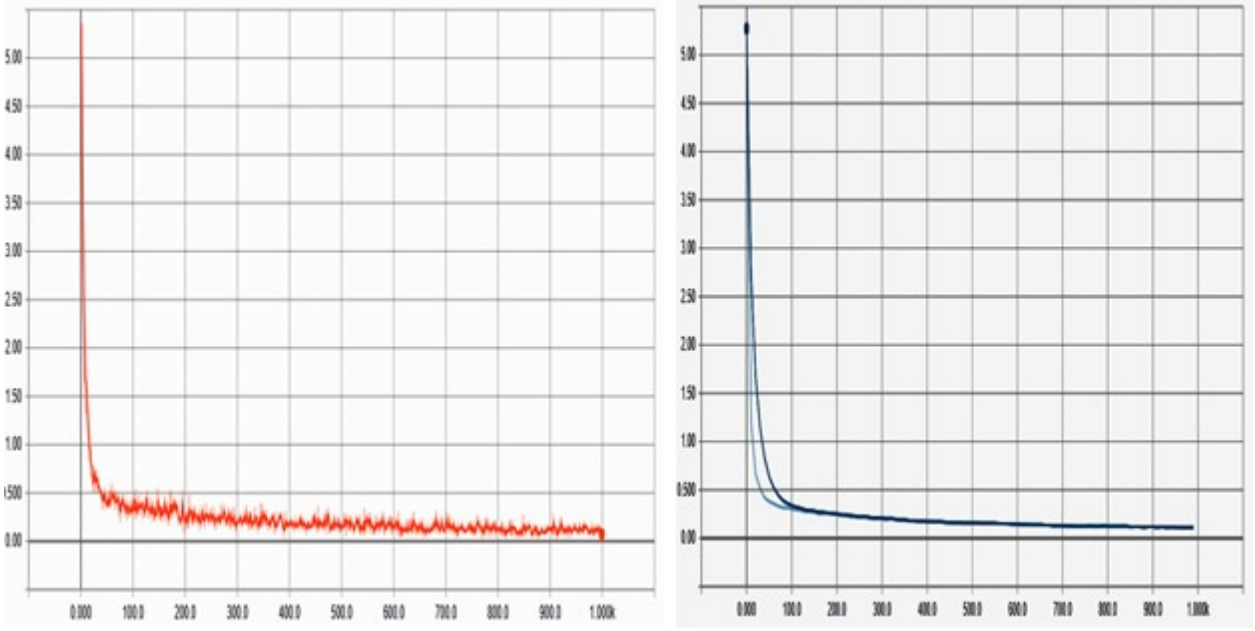

Рисунок 16 – Графік зміни втрат (sparse cross entropy) в залежності від кількості кроків: а) при тренуванні, б) при тестуванні

Однак дослідження поведінки нейронної мережі на такій моделі ускладнено, – вона працює дуже повільно.

Як вже згадувалося вище, API Tensorflow надає кілька можливостей для побудови і дослідження роботи нейромереж. Одна з них робота з Estimator (оцінювач). Так спочатку необхідно визначити конфігурацію мережі: кількість і тип шарів, алгоритми розрахунку точності і втрат, алгоритм навчання. А після, використовуючи вбудований метод тренування, просто дочекатися, коли закінчиться тренування. При цьому даний метод володіє можливістю збереження стану мережі, а також необхідним рівнем деталізації, що дає можливість аналізу етапу тестування. При цьому даний метод вже має дуже високу швидкість навчання. Приклад такої реалізації дано в Додатку 2.

У цьому модулі мережа визначається в процедурі cnn_model_fn. Тут задана згорткова нейронна мережа з двох згорткових шарів (32 фільтри і матриця виділення ознак 5x5 в першому шарі та 64 фільтри і матриця 5x5 у другому шарі), одного прихованого шару на 1024 нейрони і вихідного шару, який виділяє класи за допомогою логістичної функції (softmax). Втрати розраховуються як функція розрідженої крос-ентропії (sparse cross entropy). Тренування мережі відбувається за допомогою градієнтного спуску. Після

тренування мережі ми отримуємо: треновану мережу, стан якої записано, файл логів кожного кроку з отриманим прогнозом і розрахованими ймовірностями. Така побудова модуля дає можливість швидко змінювати конфігурацію мережі і досліджувати вплив окремих факторів на якість розпізнавання, а також будувати будь-які нейронні мережі (персептрон, багатошарові персептрони, згорткові мережі).

Однак найшвидшим за швидкодією є модель з використанням найвищого рівня API і компіляцією моделі. Приклад такої реалізації дано в Додатку 3.

У таблиці 4.4 наведені швидкості тренування двох видів моделей на різних наборах символів с однаковою структурою за одну епоху навчання.

Таблиця 4.4 – Відносна швидкість роботи моделей

| | MNIST (60000), сек. | EMNIST By class (692932), сек. | EMNIST By merge (692932), сек. |
|---|---|---|---|
| Модель с Estimator | 37 | 1374 | 1217 |
| Модель скомпільована | 5,46 | 96,32 | 96,32 |

Тому, в подальшому, для збору статистики та аналізу необхідно використовувати скомпільовані моделі.

## 4.4 Навчання нейронних мереж та аналіз статистики їх роботи

У таблиці 4.5 наведена інформація про результати тренування і тестування різних мереж.

Таблиця 4.5 – Зведена таблиця результатів тестування різних типів нейронних мереж.

| **Тип мережі** | Параметри мережі | **Точність на наборі символів** | | |
|---|---|---|---|---|
| | | MNIST | EMNIST By class | EMNIST By merge |
| Персептрон | 512 нейронів | 0.9755 | 0.8383 | 0.8727 |
| Персептрон | 1000 нейронів | 0.9772 | 0.8438 | 0.8793 |
| Персептрон | 2 шари по 512 нейронів | 0.9771 | 0.8030 | 0.8432 |
| Згорткова мережа | 32-32-1024 (3x3) | 0.9888 | 0.8597 | 0.8933 |
| Згорткова мережа | 32-32-1024 (5x5) | 0.9911 | 0.8599 | 0.8922 |
| Згорткова мережа | 64-64-1024 (5x5) | 0.991 | 0.8646 | 0.9003 |
| Згорткова мережа | 40-80-500-1000-2000 (5x5) | 0.9919 | 0.8716 | 0.9053 |

Як бачимо навіть дуже «проста» нейронна мережа показує хороші результати при класифікації з набору символів MNIST (цифри). Однак при значному розширенні набору символів EMNIST by class нейромережі починають дуже помилятися. Це, в першу чергу, пов'язано з великою кількістю символів схожих один на одного. Це однозначно підтверджується при переході на набір символів EMNIST by merge. Але так як в наборі залишаються схожі класи (i-1-7, o-0-Q-D, 6-d), то і деякі помилки залишаються.

Розглянемо більш докладно виникнення помилок в наборі символів MNIST (там помилок менше чисельно) для згорткової мережі типу 64-64-1024 (перші два шари згорткові і ще один прихований шар). Остання нейронна мережа хоч і показала найкращі варіанти розпізнавання, але вона працює в 2,5 рази повільніше і займає значно більше пам'яті (майже в 7 разів більше в порівнянні з обраною нейромережею).

При тестуванні було отримано 87 помилок розпізнавання з набору в 10000 символів, що становить точність розпізнавання 99,13%.

На рис. 17 та 18 показані символи, які були розпізнані не вірно.


Рисунок 17 – Перелік невірно розпізнаних символів, частина перша


Рисунок 18 – Таблиця невірно розпізнаних символів

Згідно з зображень 17 та 18 частину символів навіть людина не завжди адекватно може розпізнати. Інша частина символів однозначно не розпізнається, але частина символів нейромережа розпізнала неправильно з невеликою перевагою, наприклад перший символ або «нуль» або «чотири» або «шість».

Тепер розглянемо ситуацію з розпізнаванням символів набору EMNIST by class.

Тут гігантський набір тестових символів 116 323. З цієї кількості неправильно розпізналося 15750 символів, що відповідає точності 86,46%. Тут ситуація приблизно така ж, але ще більш обтяжена наявністю схожих класів символів.

Таким чином на рис. 19 можна побачити символи, які важко розпізнаються, але є не підготовлені. При ретельній обробці можна покращити шанс розпізнавання.

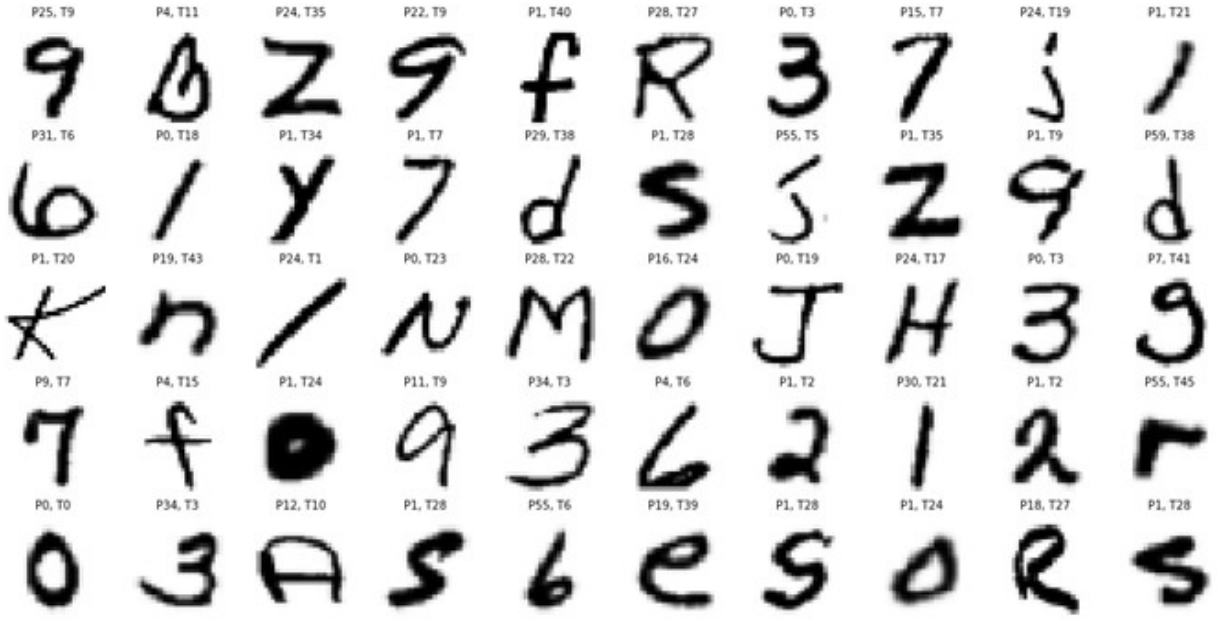


Рисунок 19 – Перелік невірно розпізнаних символів EMNIST by class (випадкова вибірка)

Для поліпшення ситуації з розпізнаванням в [11] була запропонована методика навчання згорткової мережі з використанням еластичних спотворень вхідних символів, що дало можливість покращити роботу згорткової мережі в цілому. Крім того авторами була запропонована структура згорткової мережі, яка при мінімізації розміру (а відповідно і обсягу обчислень при навчанні мережі) зберігала дуже високі показники роботи.

У цій роботі також зазначається, що частина символів тестового набору неякісні, а частина важко впізнаються навіть людиною.

Однак, так як в нашому конкретному випадку стоїть завдання розпізнавання обмеженого числа символів, які найбільш часто зустрічаються в математичних формулах, то буде кращим створити свою вибірку символів, а потім після розпізнавання проводити післяобробку з метою заміни на потрібні символи. Наприклад, якщо виключити з нашої вибірки літери O і o, то

значно покращиться розпізнання класу цифра 0. А якщо на післяобробку потрапить послідовність c0s (x ^ 2-1), післяпроцесор точно знатиме, що це косинус.

Таким чином, створюється своя вибірка, в яку входять символи EMNIST by class, але повністю виключені великі літери (так як в математичних тригонометричних і трансцендентних рівняннях вони не використовуються), а також виключені малі літери o, i, t, l, u. Таким чином отримуємо 31 клас символів для розпізнання. Виключивши з навчальної та тестової вибірки «не потрібні» символи отримуємо навчальну вибірку на 426 836 символів і тестову вибірку на 91438 символів.

Провівши навчання і тестування обраної згорткової нейронної мережі отримуємо результат – 5136 помилок. Це відповідає точності розпізнання 94,38%. Цей показник набагато вищий, ніж на вибірці EMNIST by class і незначно вищий ніж на вибірці EMNIST by merge.

Вивчимо частину коду, який відповідає за навчання нейроної мережі:

```
        # Calculate Loss (for both TRAIN and EVAL modes)
          loss  =  tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)
        # Configure the Training Op (for TRAIN mode)
        if mode == tf.estimator.ModeKeys.TRAIN:
                                                optimizer           = tf.train.GradientDescentOptimizer(learning_rate=0.001)
          train_op = optimizer.minimize(
              loss=loss,
              global_step=tf.train.get_global_step())
              return  tf.estimator.EstimatorSpec(mode=mode,  loss=loss, train_op=train_op)
```

Згідно з наведених даних, у якості функціїї оптимізації було обрано алгоритм градієнтного спуску. Також для зменьшення помилки було використано функцію sparse_softmax_cross_entropy. В інформаційних технологіях перехресна ентропвя — це число між двома розподілами ймовірностями, який вимірює кількість бітів, необхідних для ідентифікації процесу. Далі розглянемо частину програми, яка безпосередьно запускає процесс навчання:

```
# Train the model
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": train_data},
    y=train_labels,
    batch_size=100,
    num_epochs=None,
    shuffle=False)
mnist_classifier.train(
    input_fn=train_input_fn,
    steps=20000,
    hooks=[logging_hook])
```

Згідно наведених даних, для запуску алгоритма необхідно:

– Набір даних у парі: дані та очікуваний (абсолютний) результат
– Розмір пакету даних на якому відбувається навчання (batch_size)
– Кількість епох (num_epochs)
– Чи потрібно перемішати дані у випадковому порядку (shuffle)
– Функція, яка буде використовуватись для навчання (input_fn). У даному випадку це train_input_fn.
– Кількість кроків (steps).

– Хуки логів (hooks). Потрібні для детального перегляду процесу навчання.

Після виконання навчання необхідно застосувати модель. Дана операція називається Predict (передбачити). У даному контексті — розпізнати символ. Нижще наведено частину програми, яка налаштовує дану операцію:

На практиці, результати трейнування можна вивчити логи інструмента навчання. Логи, при відповідному налаштувані дозволяють переглянути процес алгоритму. Використовується для обробки технічних складнощив та аналізу результатів. Після виконання всіх циклів навчання буде згенеровано такі файли:

– mnist_err_6_1.log
– mnist_err_6c_1.log
– mnist_err_6m_1.log
– mnist_err_ln_cm_1.log
– mnist_err_ln_em_1.log
– mnist_err_ln_m_1.log

Кожен з цих файлів має дані щодо помилок розпізнавання на тестовій виборці. Згідно з назви файлу можна побачити алгоритм, який використовувався для навчання нейроної мережі (наприклад, логаріфчмічний алгорити чи лінійний).

Нижще наведено частину логу, повторювальні або неінформативні дані було опущено:

```
2021-02-07 02:35:11,246 - tensorflow - INFO - num [4699] - Test [6] - Predict [[1]]
2021-02-07 02:35:11,340 - tensorflow - INFO - num [4740] - Test [3] - Predict [[5]]
2021-02-07 02:35:15,215 - tensorflow - INFO - num [6597] - Test [0] - Predict [[7]]
2021-02-07 02:35:15,278 - tensorflow - INFO - num [6625] - Test [8] - Predict [[2]]
2021-02-07 02:35:15,340 - tensorflow - INFO - num [6651] - Test [0] - Predict [[6]]
2021-02-07 02:35:16,481 - tensorflow - INFO - num [7216] - Test [0] - Predict [[6]]
2021-02-07 02:35:17,965 - tensorflow - INFO - num [7928] - Test [1] - Predict [[0]]
2021-02-07 02:35:18,778 - tensorflow - INFO - num [8316] - Test [7] - Predict [[2]]
```

```
2021-02-07 02:35:18,965 - tensorflow - INFO - num [8408] - Test [8] - Predict [[5]]
2021-02-07 02:35:20,231 - tensorflow - INFO - num [9015] - Test [7] - Predict [[2]]
2021-02-07 02:35:21,590 - tensorflow - INFO - num [9664] - Test [2] - Predict [[7]]
2021-02-07 02:35:21,653 - tensorflow - INFO - num [9692] - Test [9] - Predict [[7]]
2021-02-07 02:35:21,731 - tensorflow - INFO - num [9729] - Test [5] - Predict [[6]]
2021-02-07 02:35:21,809 - tensorflow - INFO - num [9770] - Test [5] - Predict [[0]]
2021-02-07 02:35:22,294 - tensorflow - INFO - Errors - 81
2021-02-07 02:35:22,294 - tensorflow - INFO - 62, 247, 445, 582, 646, 659, 883, 947,
1014, 1039, 1226, 1232, 1247, 1260, 1319, 1414, 1549, 1686, 1709, 1878, 1901, 2035,
2070, 2098, 2118, 2130, 2135, 2293, 2369, 2387, 2462, 2597, 2654, 2896, 2927, 2939,
3023, 3030, 3060, 3422, 3503, 3520, 3558, 3681, 3727, 3780, 3808, 3941, 3985, 4007,
4027, 4176, 4284, 4487, 4507, 4571, 4699, 4740, 4761, 4807, 4823, 4860, 5937, 5955,
6091, 6172, 6560, 6571, 6572, 6597, 6625, 6651, 7216, 7928, 8316, 8408, 9015, 9664,
9692, 9729, 9770,
```

Згідно з даними можна виділити патерн. Даний лог містить дані які не збігаються з тестовою вибіркою. Графа Predict виводить результат, який було опрацьовано за допомогою нейроної мережі. Графа Test виводить очікуваний результат, тобто дані, які містятся у тестовій виборці. Графа num – це порядковий номер у базі даних.

Строчка

```
2021-02-07 02:35:22,294 - tensorflow - INFO - Errors – 81
```

Повідомляє о кількості елементів, які не збіглися з тестовою вибіркою. Строчкою нижче програма виводить перелік порядкових номерів, які не вдалося розпізнати для подального аналізу.

# ВИСНОВКИ

У роботі розглянуті теоретичні основи побудови штучних нейронних мереж, також були розглянуті та проаналізовані інструменти та їх аналоги. Розглянуто різні методи навчання нейронних мереж.

На практиці реалізовані і перевірені кілька видів нейронних мереж. Було проведено їх навчання і проведено тестування. За результатами тестування була перевірена гіпотеза про збільшення точності розпізнавання при зменшенні в вибірці схожих символів.

У роботі використовувався мова програмування Python і API Tensorflow для побудови і оптимізації роботи нейронних мереж.

Нейронна мережа за час навчання збирає статистичні данні, робить ймовірнісне передбачення та ймовірнісний прогноз, що також є і завданнями статистики.

В результаті роботи завдання збору статистичних даних по роботі нейронних мереж була виконана. У процесі дослідження появи помилкових відповідей нейронних мереж було зроблено висновок, що частина помилок з'являється у результаті:

- введення тестової інформації, яка апріорі не може бути розпізнаною («почерк лікарів»);
- не ідеальної роботи нейронної мережі, так як досягти врівноваженого стану нейронної мережі, особливо при схожих символах класів, дуже складно.

Навіть отримання помилки можна використовувати для поліпшення роботи системи в цілому. При отриманні помилки не однозначної, тобто, коли є декілька близьких ймовірностей результату, можна дати команду препроцесору заново відрегулювати або замінити графічні фільтри, що дасть можливість відсіяти зайвий шум у вхідних даних.

Після успішного трейнування моделі її можна зберегти та покращувати, а також опублікувати використовуючи різні підходи: локально або у ручному режимі або за допомогою хмарних технології. Наразі Google Cloud Platfom та Amazon Web Services надає можливість публікувати та адмініструвати моделі. У такому випадку вони стають доступними для доступу у мережі Інтернет.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**


1. С. Хайкин. Нейронные сети полный курс. 2-е изд., испр.: пер. с англ. – М.: ИД «Вильямс», 2006. – 1104 с.
2. Python. Вікіпедія. URL: https://ru.wikipedia.org/wiki/Python (дата звернення 08.08.2021).
3. PyCharm. Вікіпедія. URL: https://ru.wikipedia.org/wiki/PyCharm (дата звернення 08.08.2021).
4. MNIST. Вікіпедія. URL: https://ru.wikipedia.org/wiki/MNIST_(%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (дата звернення 08.08.2021)
5 Мерков А.Б. Распознавание образов. Построение и обучение вероятностных моделей - М.: ЛЕНАНД, 2014. — 240 с.
6 Гайдуков Н.П., Савкова Е.О. Обзор методов распознавания рукописного текста. Международная научно-техническая конференция студентов, аспирантов и молодых учёных "Информационно-управляющие системы и компьютерный мониторинг – 2012"
7 Ла Суан Тханг, кандидат технических наук, Диссертация: Методы распознавания рукописных текстов в системах автоматизации документооборота на промышленных предприятиях, http://www.dissercat.com/content/metody-raspoznavaniya-rukopisnykh-tekstov-v-sistemakh-avtomatizatsii-dokumentooborota-na-pro#ixzz5XiYWqZnK
8 Мерков А.Б. Распознавание образов. Введение в методы статистического обучения – «Едиториал УРСС», 2006. – 256 с.
9 Каллан Роберт. Основные концепции нейронных сетей. пер. с англ. — М. : ИД "Вильямс", 2001. — 287 с
10 Qiao, Yu THE MNIST DATABASE of handwritten digits (http://www.gavo.t.u-tokyo.ac.jp/~qiao/database.html)

11 LeCun, Yann MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges (http://yann.lecun.com/exdb/mnist/)

12 The EMNIST Dataset (https://www.nist.gov/itl/iad/image-group/emnist-dataset)

13 EMNIST: an extension of MNIST to handwritten letters. Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andr´e van Schaik The MARCS Institute for Brain, Behaviour and Development Western Sydney University (https://arxiv.org/abs/1702.05373v1).

14 Солдатова О.П., Гаршин А.А. Применение сверточной нейронной сети для распознавания рукописных цифр. Компьютерная оптика, том 34, №2, 2010

15 Барковський В. В., Барковська Н.В., Лопатін О.К. Теорія імовірностей та математична статистика, 5-те видання, Київ: “Центр учбової літератури”, 2010. – 424 с.

16 ГМУРМАН В.Е., Теория вероятностей и математическая статистика. Учеб. пособие для вузов/В. Е. Гмурман. — 9-е изд., стер. —. М.: Высш. шк., 2003. – 479 с.

17 Руденко В. М., Математична статистика. Навч. посіб. – К.: Центр учбової літератури, 2012. – 304 с.