МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра інформаційних систем
(повна назва кафедри)

ДИПЛОМНА РОБОТА

СИСТЕМА ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав(ла): студент(ка) групи ПМІ-44
спеціальності 122 – комп'ютерні науки
(шифр і назва спеціальності)

Ковалів В. В.
(підпис) (прізвище та ініціали)

Керівник Горлач В. М.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2023

ЗМІСТ

ВСТУП **3**

1. Поняття нейронної мережі 3
2. Постановка задачі 4

Розділ 1 ЗАГАЛЬНА СТРУКТУРА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ **7**

1.1 Поняття згорткової нейронної мережі 7

1.2 Обробка за допомогою фільтру 8

1.3 Загальний вигляд 11

РОЗДІЛ 2 ПОБУДОВА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ **14**

2.1 Інсталяція необхідних бібліотек 14

2.2 Імпорт даних 15

2.3 Дослідження характеристик даних 17

2.4 Створення моделі 20

2.5 Тренування моделі 21

2.6 Результат 23

ВИСНОВОК **26**

Посилання та література **27**

ВСТУП

1. Поняття нейронної мережі

Для розуміння того, що таке штучні нейронні мережі і як вони функціонують, слід мати уявлення про принципи роботи людського мозку. Мозок містить нейрони - органічні перемикачі, які змінюють тип переданих сигналів в залежності від електричних або хімічних сигналів, що в них передаються.

Нейронна мережа в людському мозку - це складна система зв’язків між нейронами, де сигнал, переданий одним нейроном, може поширюватись до тисяч інших нейронів. Навчання відбувається шляхом посилення певних нейронних зв’язків через повторну активацію, що сприяє збільшенню ймовірності виведення потрібного результату при відповідному вхідному сигналі. У такому виді навчання за допомогою зворотного зв'язку, при досягненні правильного результату, нейронні зв'язки, що сприяють його формуванню, зміцнюються і стають більш міцними.

Штучні нейронні мережі моделюють простішу версію мозку і можуть бути навчені за допомогою контрольованих або неконтрольованих методів. У контрольованому навчанні мережа отримує вхідну інформацію та відповідні приклади вихідної інформації для навчання. Наприклад, допустимо, є деякий магазин, який хоче прогнозувати продажі своїх товарів наступного місяця. Вхідною інформацією для мережі можуть бути дані про кількість продажів кожного товару протягом останніх місяців, ціни на товари, рекламні витрати на продукти тощо. Вихідна інформація може бути кількість продажів для кожного товару, що очікується наступного місяця. Мережа навчається на історичних даних і розраховує прогнози продажу на майбутній місяць на основі вхідної інформації.

В нейронних мережах, що моделюють біологічні нейрони, для передачі сигналів використовується активаційна функція. У випадках класифікації, таких як визначення спаму, важливо, щоб активаційна функція мала властивість "перемикача", тобто якщо вхідний сигнал перевищує певне значення, вихід повинен змінити стан з 0 на 1 або з -1 на 1, і цим самим імітувати роботу біологічних нейронів. Для цього часто використовується сигмоїдна функція, яка

дозволяє виконати цю операцію швидко та ефективно. Сигмоїдна функція має форму "S"-подібної кривої та вона використовується для обмеження значень вихідного сигналу в межах від 0 до 1. Коли вхідний сигнал досягає певного порогового значення, сигмоїдна функція змінює свій вихідний сигнал з 0 на 1, що дозволяє здійснити розпізнавання об'єктів та класифікацію даних.

Біологічні нейрони утворюють ієрархічну мережу, де одні нейрони отримують вхідний сигнал від інших нейронів та генерують вихідний сигнал, який слугує вхідним сигналом для інших нейронів. Ця ієрархічна організація нейронів дозволяє створювати складні поведінкові патерни та реалізовувати різноманітні когнітивні функції.

Моделюючи цю біологічну мережу, ми можемо представити її у вигляді зв'язаного шару з вузлами. Кожен вузол отримує зважені вхідні дані, які використовуються для обчислення вихідного значення вузла. Вузли можуть містити різні активаційні функції, які підсумовують вхідні дані та генерують вихідний сигнал.

Зважені дані, які надходять до вузла, обчислюються шляхом множення вагових коефіцієнтів на вхідні дані та їх підсумовування в вузлі. Вагові коефіцієнти є числовими значеннями, які можуть змінюватись протягом процесу навчання. Вагові коефіцієнти є важливим параметром нейронної мережі, оскільки вони визначають вплив вхідних даних на вихідні значення вузла.

Крім вагових коефіцієнтів, вузол може містити вагу елемента зміщення. Це дозволяє вузлу стати більш гнучким та додатково збільшує його вплив на вихідний сигнал. Елемент зміщення додається до вагових коефіцієнтів під час обчислення вихідного значення вузла.

*2. **Постановка задачі***

Прийняття рішень в медицині є складним процесом, який вимагає урахування багатьох факторів. Труднощі виникають коли різних чинників, які потрібно брати до уваги, стає все більше і більше.

Машинне навчання та штучний інтелект мають великий потенціал для застосування в медицині. Штучний інтелект займає провідну позицію в цій галузі,

а машинне навчання є важливою складовою його функціонування. Машинне навчання дозволяє комп'ютерам самостійно вчитися, але цей процес вимагає попередньої підготовки від людей.

Автоматизована класифікація – дуже активна сфера досліджень. Зокрема, використання медичних зображень для створення швидкої й точної класифікації може заощадити час, зусилля і витрати, а також зменшити помилки. Задача класифікації полягає у формалізованій процедурі, що включає множину об'єктів, які поділяються на класи, до яких належать об'єкти вибірки, а класифікація об'єкта полягає у визначенні його класу. В машинному навчанні, задачу класифікації зазвичай розв'язують за допомогою методів штучної нейронної мережі. Люди навіть без свідомого зусилля можуть швидко та легко розпізнавати об'єкти та їх характеристики, що оточують нас. Однак, комп'ютерам важко опанувати навички швидкого розпізнавання шаблонів та узагальнення знань. Використання машинного навчання та штучного інтелекту може допомогти збільшити точність та ефективність класифікації об'єктів, включаючи медичні зображення.

Однак машинне навчання не замінює роль лікарів та експертів у медичній діагностиці. Застосування машинного навчання у медицині може служити як допоміжний інструмент для експертів, які знають контекст та медичні проблеми. Лікарі та медичні експерти можуть використовувати результати класифікації, отримані за допомогою машинного навчання, для підтвердження або спростування своїх гіпотез та діагнозів.

Один з прикладів застосування машинного навчання для класифікації зображень є класифікація гістологічних зрізів колоректального раку. Створення нейронної мережі для класифікації гістологічних зображень допоможе швидко та ефективно аналізувати великі обсяги даних та зменшити ризик помилок, пов'язаних з людським фактором. Модель може виявляти відмінності між зображеннями та класифікувати їх у відповідні категорії з більшою точністю, ніж людина. Це може допомогти покращити діагностику та лікування пацієнтів з колоректальним раком.

Ось чому є актуальним розробка нейронної мережі для класифікації зображень в гістології колоректального раку, оскільки машинне навчання може допомогти збільшити точність та ефективність діагностики.

Розділ 1 ЗАГАЛЬНА СТРУКТУРА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

1.1 Поняття згорткової нейронної мережі

Згорткова нейронна мережа (Convolutional Neural Network або CNN) - це спеціалізований тип штучної нейронної мережі, який призначений для обробки даних з просторовою структурою, зокрема зображень. Вона використовує фільтри, що просуваються по вхідному зображенню, для виявлення різних шаблонів та ознак.

Основна ідея згорткових нейронних мереж полягає у використанні фільтрів, які просуваються по вхідному зображенні або сигналу, здійснюючи операцію згортки. Згортка полягає у взятті добутку елементів фільтра і вхідних значень, які перекриваються, і підсумовує їх для отримання вихідного значення. Це дозволяє виявляти локальні шаблони або ознаки в зображенні.

Основні компоненти згорткової нейронної мережі включають:

1. Шари згортки (Convolutional layers): ці шари містять фільтри, які застосовуються до вхідних даних для виявлення локальних шаблонів. Кожен фільтр є невеликим окремим набором ваг, який переміщується по вхідному зображенню або карті ознак і виконує операцію згортки. Процес згортки полягає у взятті добутку елементів фільтра та відповідних пікселів вхідного зображення, які перекриваються. Результатом цієї операції є одне значення, яке представляє наявність або інтенсивність певної ознаки у даному місці.

2. Шари пулінгу (Pooling layers): ці шари використовуються для зменшення розміру вихідних карт ознак, зберігаючи основні ознаки й знижуючи кількість параметрів моделі. Це допомагає покращити обчислювальну ефективність моделі та зменшити ризик перенавчання. Операція пулінгу проводиться над кожним окремим каналом (ознакою) вихідної карти ознак. Під час пулінгу використовуються фіксовані фільтри (наприклад, 2x2 або 3x3) для переміщення по вхідній карті ознак. На кожному кроці фільтр вибирає максимальне або середнє значення у певній області, яку він охоплює.

3. Повнозв'язані шари (Fully Connected layers): після декількох шарів згортки та пулінгу, зазвичай використовуються повнозв'язані шари, щоб об'єднати отримані ознаки та здійснити кінцеву класифікацію або регресію. Одна з важливих властивостей повнозв'язаних шарів полягає в їх здатності до вивчення неоднорідних та нелінійних залежностей між ознаками. Це означає, що модель може виявити складні шаблони та взаємозв'язки, які можуть бути важкими для виявлення у просторі ознак. Завдяки цьому, модель здатна досягти високої точності та прогностичної здатності.

Основні шари згорткової нейронної мережі зображені на рисунку 1.1

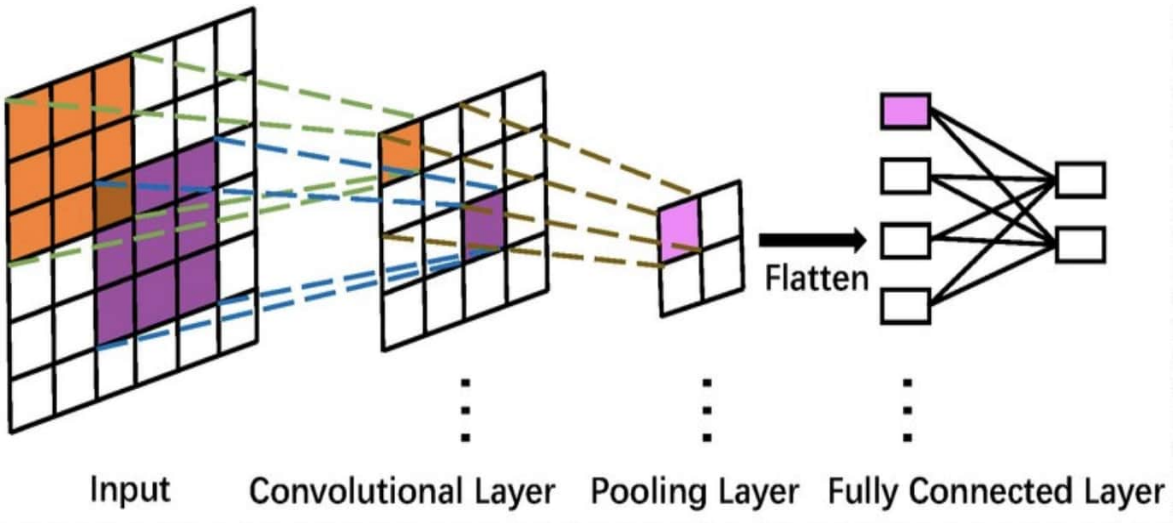

Рис. 1.1 Основні компоненти CNN

Згорткові нейронні мережі є потужним інструментом у галузі комп'ютерного зору та аналізу даних з просторовою структурою. Вони широко використовуються для визначення образів, розпізнавання об'єктів, обробки зображень, аналізу природних мов та багатьох інших завдань, пов'язаних з обробкою даних, що мають геометричну або просторову природу.

1.2 Обробка за допомогою фільтру

Перша обробка зображення за допомогою вертикального та горизонтального граничних фільтрів використовується для виявлення границь об'єктів на зображенні. Кожен з цих фільтрів є математичним шаблоном, який застосовується

до кожного пікселя зображення для визначення його внутрішнього контурного відтінку.

Вертикальний граничний фільтр, який використовується для першої обробки зображення, є математичним шаблоном або ядром, який дозволяє виділити вертикальні границі на зображенні. Він застосовується шляхом сканування зображення зліва направо, розміщуючи шаблон фільтра на кожній позиції й обчислюючи суму елементів, що збігаються з пікселями зображення. Результат цієї операції надає оцінку наявності вертикальної границі. Після застосування вертикального граничного фільтра до зображення ми отримуємо нове зображення, в якому вертикальні границі об'єктів стають більш помітними.

Розглянемо сіре зображення, представлене матрицею розміром 6x6, де кожен елемент матриці відображає інтенсивність пікселя в відповідній точці зображення. Щоб виявити вертикальні границі на даному зображенні, використаємо згорткову операцію, яка зводиться до обчислення суми поелементного добутку у вікні, яке пересувається по всьому зображенню. (рис. 1.2)

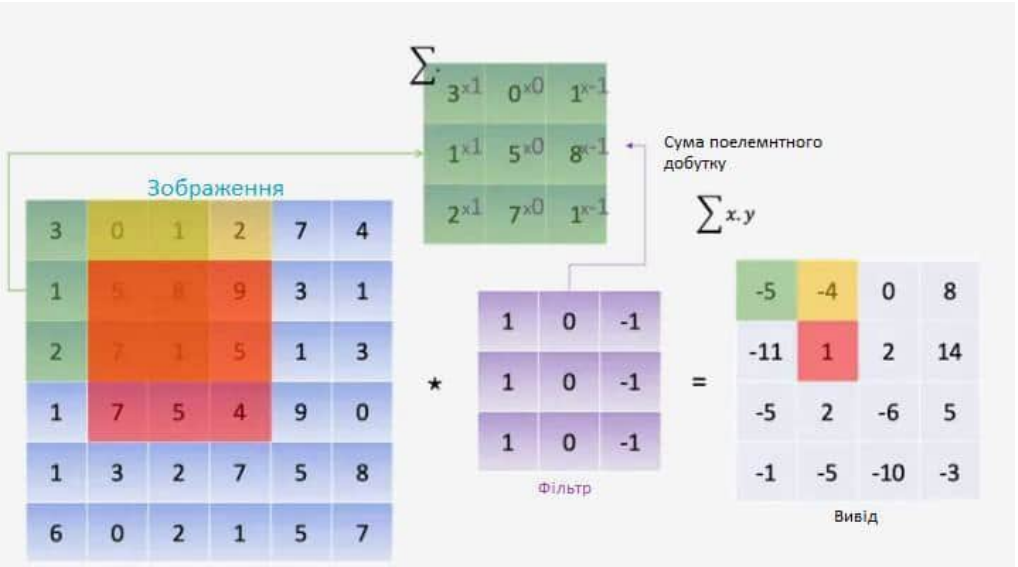

Рис. 1.2 Схема обчислення суми поелементного добутку

Проводимо поелементне множення в першому 3x3 блоку, а потім переходимо до наступного блоку справа і повторюємо ту ж операцію. Процес повторюється, доки не пройдемо всі можливі блоки. Цей процес зображений на рис. 1.3

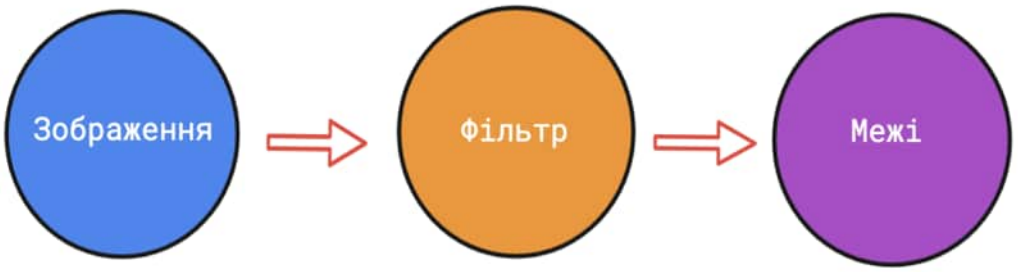

Рис. 1.3 Процес пропускання через фільтр

Основна ідея полягає в тому, що нейронна мережа отримує зображення як вхідний сигнал і застосовує різні обчислювальні процеси для отримання певного результату. Вхідне зображення може бути подане у вигляді двовимірної матриці пікселів, а нейронна мережа може використовувати свої внутрішні ваги та алгоритми для обробки цього зображення.

Отримання певного результату залежить від завдання, яке передбачено для нейронної мережі. Це може бути класифікація зображень, сегментація областей, визначення атрибутів або будь-яке інше завдання, пов'язане з обробкою зображень. Нейронна мережа використовує вхідні сигнали, у нашому випадку зображення, та її внутрішні параметри, щоб виконати необхідні обчислення та забезпечити відповідний результат.

Наприклад, нейронна мережа може проводити розпізнавання об'єктів на зображенні. Вона може використовувати ваги нейронів, для виявлення особливостей і залежностей у зображенні. Це може допомогти під час класифікації образів.(рис. 1.4)

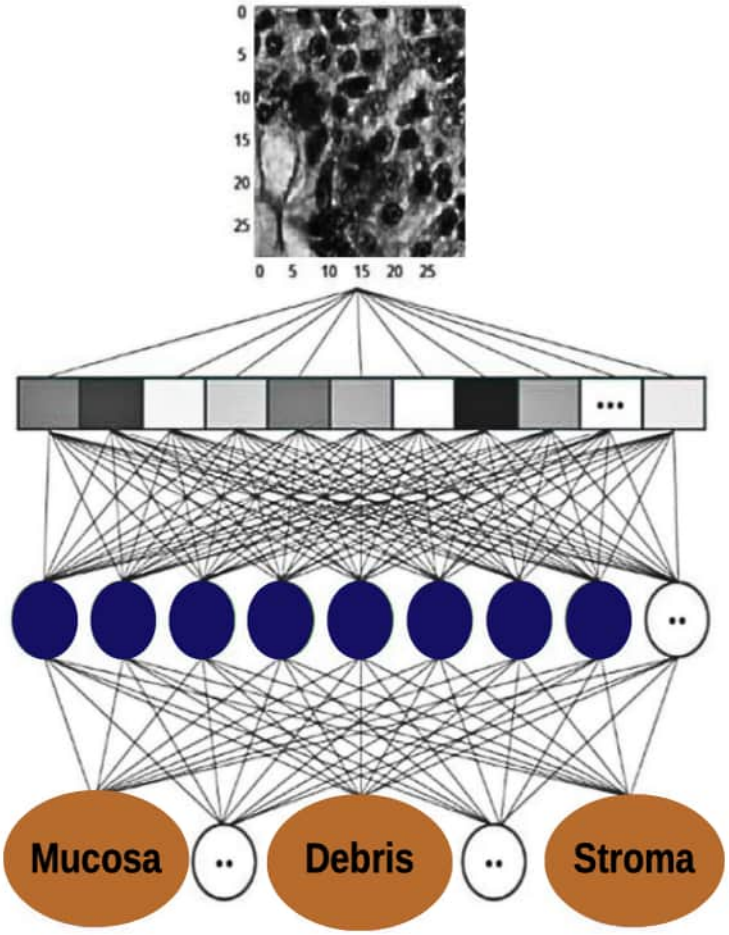

Рис. 1.4 Схема класифікації зображення

1.3 Загальний вигляд

Згорткові нейронні мережі складаються з послідовності операцій, які виконуються над вхідними даними. Основними операціями в CNN є згортка (convolution), нелінійна активація (non-linear activation), і пулінг (pooling).(рис 1.5)

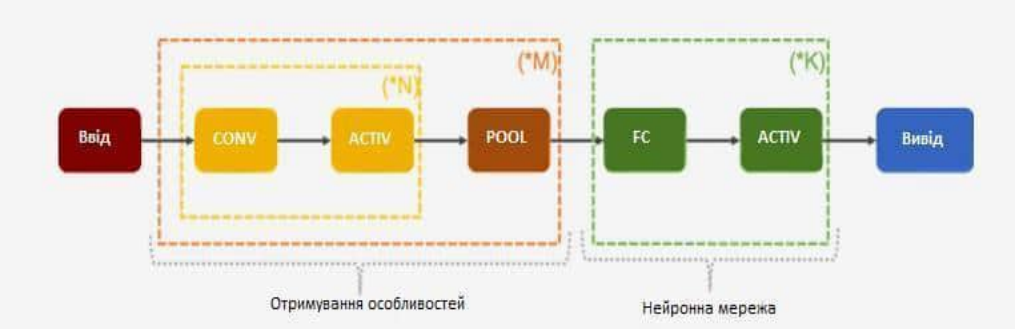

Рис. 1.5 Загальний вигляд CNN

Після певної кількості згорткових шарів і застосування активаційної функції, застосовується процес об'єднання (пулінг), який допомагає зменшити розмірність отриманого вихідного зображення. Після цього цей процес повторюється певну кількість разів.

Ці операції, такі як згортка, активаційна функція та об'єднання, дозволяють виявити та виділити особливості зображення, такі як границі, текстури або форми. Вони створюють нові представлення зображення, які передаються у зв'язувальні шари (повнозв'язні шари) нейронної мережі. Кожен зв'язувальний шар супроводжується активаційною функцією, яка додає нелінійність до вихідних значень та допомагає моделі здійснювати складніші обчислення та виконувати розпізнавання або класифікацію зображень.

Отримані особливості та інформація з зображення передаються через зв'язувальні шари, де відбувається подальше оброблення та прийняття рішень. Цей процес, який включає згорткові шари, об'єднання, зв'язувальні шари та активаційні функції, допомагає нейронній мережі виявляти складні залежності й робити високоякісні прогнози на основі вхідних зображень.

У тривимірному вигляді згорткову нейронну мережу можна представити наступним чином: (рис. 1.6)

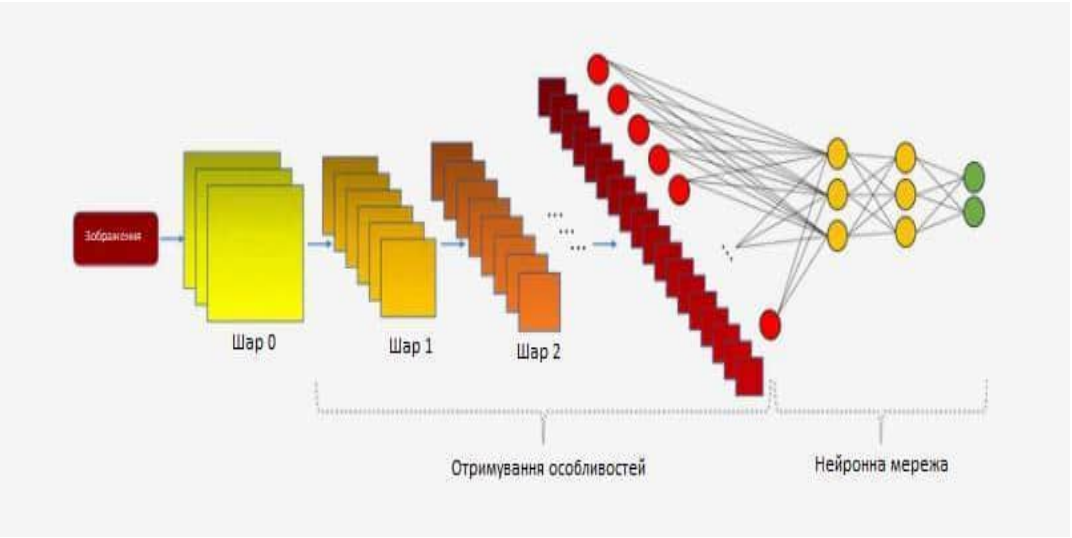

Рис. 1.6 Схема CNN у тривимірному вигляді

Процес згорткової нейронної мережі в тривимірному вигляді може повторюватись з кількома шарами згортки та об'єднання, що дозволяє моделі виявляти все складніші особливості та ієрархічні залежності в даних.

РОЗДІЛ 2 ПОБУДОВА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Інсталяція необхідних бібліотек

Для того, щоб почати розробку CNN для класифікації текстур в гістології колоректального раку потрібно встановити певні бібліотеки, які надають інструменти для роботи з даними та їх візуалізації.

В проєкті були використані бібліотеки math та numpy. Math містить математичні функції, які дозволяють виконувати різноманітні обчислення. Numpy забезпечує зручну роботу з числовими даними у вигляді масивів та матриць, що дозволяє швидко та ефективно аналізувати та обробляти такі дані.

Для створення та навчання моделі було використано TensorFlow, що є відкритим програмним забезпеченням та TensorFlow Datasets - це набір даних, який спрощує завантаження та доступ до наборів даних з різних джерел.

Додатково, було встановлено бібліотеки, які допоможуть у подальшій роботі з даними та реалізації моделі. Наприклад, matplotlib - це бібліотека для створення графіків та візуалізації даних. Logging - це модуль, який дозволяє записувати повідомлення про стан програми для подальшого аналізу. Крім того, було використано бібліотеку tqdm, яка дозволяє відстежувати прогрес під час навчання моделі, що є корисною функцією для великих наборів даних.

Використання цих бібліотек допоможе зекономити час та зробить розробку більш продуктивною. (рис. 2.1)

```
# імпортуємо бібліотеку для математичних обчислень
import math
# імпортуємо бібліотеку для роботи з масивами даних
import numpy as np

# імпортуємо TensorFlow
import tensorflow as tf
# імпортуємо набір даних TensorFlow
import tensorflow_datasets as tfds

# імпортуємо бібліотеку для створення прогрес-бару
import tqdm.auto
# імпортуємо бібліотеку для візуалізації даних
import matplotlib.pyplot as plt
# імпортуємо бібліотеку для логування подій
import logging
```

Рис. 2.1 Інсталяція необхідних бібліотек

2.2 Імпорт даних

В даному проєкті використовується набір даних Colorectal histology, який містить 5000 зображень в 8 категоріях. Ці категорії описують типи тканин, які можна знайти на гістологічних зрізах. Колекція даних містить зображення гістологічних зрізів тканин колоректального раку, а також підписи, які показують, які типи клітин і тканин присутні на кожному зрізі. У наборі даних є 8 категорій: "Tumor", "Stroma", "Complex", "Lympho", "Debris", "Mucosa" і "Adipose", а також категорія "Empty", яка відповідає порожньому простору без клітин або тканин. Звертання до даного набору даних було реалізовано через TensorFlow використовуючи API. (рис. 2.2)

```
# Завантаження даних набору "colorectal_histology" з TensorFlow Datasets (TFDS).
# Аргумент `as_supervised=True` означає, що дані повертаються у форматі `(input, label)` (для навчання моделі).
# Аргумент `with_info=True` дозволяє отримати метадані про набір даних (кількість прикладів, розмір зображень тощо).
dataset, metadata = tfds.load('colorectal_histology', as_supervised=True, with_info=True)
```

Рис. 2.2 Імпорт даних

Для поділу даних на тренувальний та тестовий набори, використовується зазвичай стандартна пропорція 80:20 або 70:30, де 80% або 70% даних відводяться для тренування, а 20% або 30% - для тестування.

У цьому проєкті, було обрано пропорцію 80:20, що означає, що 80% даних буде використовуватись для навчання моделі, а 20% - для тестування. Кількість тестових екземплярів встановлено 700, а кількість тренувальних екземплярів - 4300. Після розділення даних, обидва набори даних були збережені в змінних. (рис. 2.3) Завдяки цьому, модель буде навчатись на тренувальних даних та тестуватись на тестових даних, що дозволить оцінити ефективність моделі на реальних даних.

```
# Встановлюємо кількість тренувальних та тестових екземплярів.
num_train_examples = 4300
num_test_examples = 700

# Беремо перші num_test_examples з train датасету для тестування моделі
test_dataset = dataset['train'].take(num_test_examples)

# Беремо num_train_examples екземплярів після num_test_examples для тренування моделі
train_dataset = dataset['train'].skip(num_test_examples).take(num_train_examples)

# Друк кількості тренувальних та тестових екземплярів
print(f'Кількість прикладів для тренування моделі: {num_train_examples}')
print(f'Кількість прикладів для оцінки її точності: {num_test_examples}')
```

Рис. 2.3 Розділення даних на тестові та тренувальні

Після завантаження та розділення набору даних отримуємо імена та мітки класів. Мітки - це масив цілих чисел в інтервалі від 0 до 9, які відповідають класу зображення. (рис. 2.4)

```
# Отримуємо імена класів та їх мітки
class_names = metadata.features['label'].names

# Виводимо імена та мітки класів
print(f'Мітки та імена класів:')
for i, class_name in enumerate(class_names):
    print(f"Class {i}: \t{class_name}")
```

Рис. 2.4 Отримання імен та міток класів.

Для зручності було виведено імена класів на екран, присвоюючи їм порядкові номери. (рис. 2.5)

```
Мітки та імена класів:
Class 0:        tumor
Class 1:        stroma
Class 2:        complex
Class 3:        lympho
Class 4:        debris
Class 5:        mucosa
Class 6:        adipose
Class 7:        empty
```

Рис. 2.5 Вивід імен та міток класів

2.3 Дослідження характеристик даних

Значення кожного пікселя в зображенні, яке отримуємо з набору даних, може мати значення в інтервалі від 0 до 255. Однак, щоб дана модель працювала коректно, необхідно привести ці значення до діапазону від 0 до 1, що відповідає нормалізованому вигляду даних. Для цього було використано функцію нормалізації,

яка ділить кожен елемент на 255, перетворює їх у значення з плаваючою крапкою та змінює діапазон значень на [0, 1].

Для того, щоб застосувати функцію нормалізації до тренувального та тестового наборів, було використано функцію map, яка дозволяє застосовувати задану функцію до кожного екземпляра окремо.(рис. 2.6)

```
def image_normalization(input_images, input_labels):
    # Перетворюємо вхідні зображення на тип float32
    # Нормалізуємо значення пікселів до діапазону [0, 1]
    normalized_images = tf.cast(input_images, tf.float32) / 255.0

    # Повертаємо нормалізовані зображення та мітки
    return normalized_images, input_labels

# Застосовуємо функцію нормалізації до тренувального та тестового датасетів
train_dataset = train_dataset.map(image_normalization)
test_dataset = test_dataset.map(image_normalization)
```

Рис. 2.6 Нормалізація даних

Також було створено функцію для відображення 16 зображень з тренувального набору даних. (рис. 2.7) Для кожного зображення відображається інформація про клас, до якого воно належить. (рис. 2.8) Це дає можливість зрозуміти, як модель працює і чи може вона правильно класифікувати зображення відповідно до їх класу. Якщо модель правильно працює, то під зображенням буде напис, який збігається з фактичним класом зображення. Для виведення зображень використовується бібліотека matplotlib.

```
# Встановлення коліру фону графіків
plt.rcParams['axes.facecolor'] = '#1E1E1E'
# Створення фігури
plt.figure(figsize=(10, 10))

# Цикл по 16 зображеннях з тестового датасету
i = 0
for (image, label) in test_dataset.take(16):
    image = image.numpy().reshape((150, 150, 3))
    ax = plt.subplot(4, 4, i + 1)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.grid(False)
    ax.imshow(image, cmap=plt.cm.binary)
    ax.set_xlabel(class_names[label], color='black')
    ax.spines['left'].set_linewidth(2)
    ax.spines['left'].set_color('white')
    ax.spines['right'].set_linewidth(2)
    ax.spines['right'].set_color('white')
    ax.spines['top'].set_linewidth(2)
    ax.spines['top'].set_color('white')
    ax.spines['bottom'].set_linewidth(2)
    ax.spines['bottom'].set_color('white')
    i += 1

# Відображаємо зображення
plt.tight_layout()
plt.show()
```

Рис. 2.7 Функція для відображення зображень

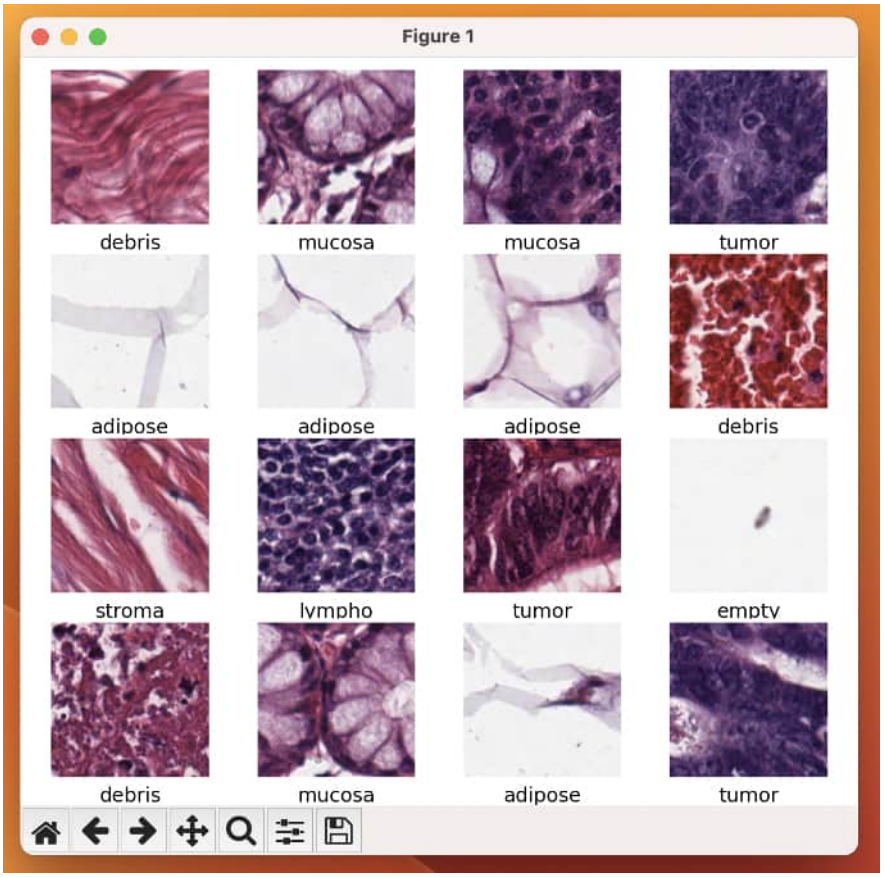

Рис. 2.8 Вивід зображень

2.4 Створення моделі

Модель нейронної мережі було створено за допомогою бібліотеки TensorFlow. Шаблон складається з трьох шарів: вхідного шару, прихованого шару та вихідного шару. (рис. 2.9)

Вхідний шар призначений для згладжування вхідних даних, що представляють зображення. Вхідні зображення мають розмір 150x150 та кольорову палітру RGB, тому вхідний шар має вхідну форму (150, 150, 3).

Прихований шар є повнозв'язним шаром зі 128 нейронами та функцією активації relu. Даний шар отримує вхідні дані зі згладженого вхідного шару та оброблює їх за допомогою матричної операції множення. Результатом роботи цього шару є вектор прихованих функцій.

Останній шар також є повнозв'язним шаром, але з 8 нейронами, що відповідають кількості класів у вихідному наборі даних. Вихідний шар

використовує функцію активації softmax, яка вирішує проблему класифікації: вибір класу, до якого належить зображення.

```
# Створення моделі нейронної мережі
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(150, 150, 3)),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(8, activation=tf.nn.softmax)
])
```

Рис. 2.9 Створення моделі нейронної мережі

Після створення моделі потрібно її скомпілювати, щоб вказати, як саме вона буде навчатися. Компіляція моделі включає вибір функції втрат, оптимізатора та метрик для оцінки результатів моделі. (рис. 2.10)

У цьому проєкті було використано Adam optimizer, що є одним з найбільш популярних алгоритмів оптимізації в глибокому навчанні. Як функція втрат обрана sparse_categorical_crossentropy, яка використовується для задач класифікації з більш ніж двома категоріями. Для вимірювання точності моделі у класифікації зображень використовується метрика accuracy, яка вказує на відсоток правильно класифікованих зображень у відношенні до загальної кількості зображень.

```
# Компілювання моделі нейронної мережі
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Рис. 2.10 Компілювання моделі нейронної мережі

2.5 Тренування моделі

Реалізація тренування моделі нейромережі складається з декількох етапів. (рис. 2.11)

Спочатку із задавання розміру пакета даних (BATCH_SIZE) та кількість епох (EPOCHS), які будуть використовуватися для тренування моделі. Наступним кроком

є підготовка даних до тренування. Створюються два набори даних - тренувальний та тестовий. Тренувальний набір повторюється за допомогою функції repeat, щоб під час тренування кожен приклад даних був використаний більше одного разу, перемішується та ділиться на пакети. Тестовий датасет тільки ділиться на пакети.

Далі із виклику метода fit, що розпочинає процес тренування. Під час кожної епохи модель переглядає всі приклади в тренувальному наборі даних та оновлює свої ваги, щоб зменшити значення функції втрат на навчальному датасеті.

Після закінчення тренування, викликається метод evaluate для обчислення точності на тестовому датасеті. В даний метод передається тестовий датасет, а також кількість кроків, які має виконати модель під час оцінки точності на тестовому датасеті.

Після закінчення обчислення точності результат тренування виводиться в консоль. Точність: 80% (рис. 2.12)

```
# Задання розміра пакета даних
BATCH_SIZE = 32
# Задання кількісті епох
EPOCHS = 78

# Підготування даних до тренування
train_dataset = train_dataset.repeat().shuffle(num_train_examples).batch(BATCH_SIZE)
test_dataset = test_dataset.batch(BATCH_SIZE)

# Процес тренування
model.fit(train_dataset, epochs=EPOCHS, steps_per_epoch=math.ceil(num_train_examples / BATCH_SIZE))

# Оцінювання точності моделі на тестовому датасеті
test_loss, test_accuracy = model.evaluate(test_dataset, steps=math.ceil(num_test_examples / BATCH_SIZE))

# Вивід точність на тестовому датасеті
print("Точність на тестовому наборі даних: ", test_accuracy)
```

Рис. 2.11 Тренування моделі

```
Epoch 68/78
135/135 [==============================] - 27s 200ms/step - loss: 0.6508 - accuracy: 0.7292
Epoch 69/78
135/135 [==============================] - 27s 203ms/step - loss: 0.5182 - accuracy: 0.7951
Epoch 70/78
135/135 [==============================] - 27s 198ms/step - loss: 0.5464 - accuracy: 0.7766
Epoch 71/78
135/135 [==============================] - 27s 200ms/step - loss: 0.4871 - accuracy: 0.7977
Epoch 72/78
135/135 [==============================] - 27s 199ms/step - loss: 0.5033 - accuracy: 0.7949
Epoch 73/78
135/135 [==============================] - 27s 203ms/step - loss: 0.4487 - accuracy: 0.8266
Epoch 74/78
135/135 [==============================] - 26s 196ms/step - loss: 0.4926 - accuracy: 0.7961
Epoch 75/78
135/135 [==============================] - 26s 196ms/step - loss: 0.4808 - accuracy: 0.8025
Epoch 76/78
135/135 [==============================] - 27s 197ms/step - loss: 0.4891 - accuracy: 0.7956
Epoch 77/78
135/135 [==============================] - 37s 271ms/step - loss: 0.4348 - accuracy: 0.8201
Epoch 78/78
135/135 [==============================] - 27s 201ms/step - loss: 0.4576 - accuracy: 0.8134
22/22 [==============================] - 2s 67ms/step - loss: 0.6026 - accuracy: 0.8014
Точність на тестовому наборі даних:  0.801428554058075
```

Рис. 2.12 Результат тренування

2.6 Результат

Для відображення результатів роботи моделі було створено дві функції - draw_picture та draw_preds. (рис. 2.13) Функція draw_picture призначена для відображення вхідних зображень та їх передбачень на графіку. Вона отримує індекс зображення, передбачення моделі для цього зображення, правильну мітку класу та масив зображень. Функція відображує зображення та його передбачення на графіку, присвоюючи правильно передбаченим зображенням зелений колір, а неправильно передбаченим - червоний. Назва графіка складається з назви класу, який було передбачено моделлю з його відповідною ймовірністю, а також правильної мітки класу.

Функція draw_preds призначена для візуалізації передбачень моделі на зображенні з індексом. Вона отримує індекс зображення, передбачення моделі для цього зображення та правильну мітку класу. Функція відображає на графіку вертикальну стовпчасту діаграму, де на горизонтальній осі зображено індекс класу (0-7), а на вертикальній - відповідну ймовірність передбачення моделі для кожного класу. Кожен стовпчик має колір, що відповідає його класифікації: зелений колір -

для правильної класифікації, червоний - для неправильної класифікації. Під графіком вказується відсоток впевненості моделі у цьому, що вхідне зображення відповідає класу, який було передбачено моделлю.

```
# Функція для відображення діаграми передбачень
def draw_preds(i, preds_array, true_label, ax):
    preds_array, true_label = preds_array[i], true_label[i]
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    thisplot = ax.bar(range(8), preds_array, color="#544848")
    ax.set_ylim([0, 1])
    predicted_label = np.argmax(preds_array)
    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('green')

# Функція для відображення зображення та мітки з передбаченнями
def draw_picture(i, preds_array, true_labels, images, ax):
    preds_array, true_label, img = preds_array[i], true_labels[i], images[i]
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])

    ax.imshow(img[..., 0], cmap=plt.cm.binary)
    predicted_label = np.argmax(preds_array)
    if predicted_label == true_label:
        color = 'green'
    else:
        color = 'red'

    ax.set_xlabel("{} {:2.0f}% ({})"
    .format(class_names[predicted_label], 100 * np.max(preds_array),
            class_names[true_label]), color=color)
```

Рис. 2.13 Допоміжні функції

Після створення допоміжних функцій відмальовується графічний інтерфейс для відображення результатів роботи нейронної мережі на тестовому наборі даних.

Спочатку створюється графік з 5 рядків та 2 стовпців ітеруючись по кожному елементу тестового набору даних. На кожному графіку відображається зображення з тестового набору, його передбачення моделі та правильна мітка класу. Після чого, з тестового набору даних витягуються зображення та правильні мітки, а потім використовуючи натреновану модель, проводиться передбачення на тестових даних.

Далі, за допомогою двох функцій draw_picture та draw_preds передбачення моделі для кожного зображення відображаються на графіку. (рис. 2.14)

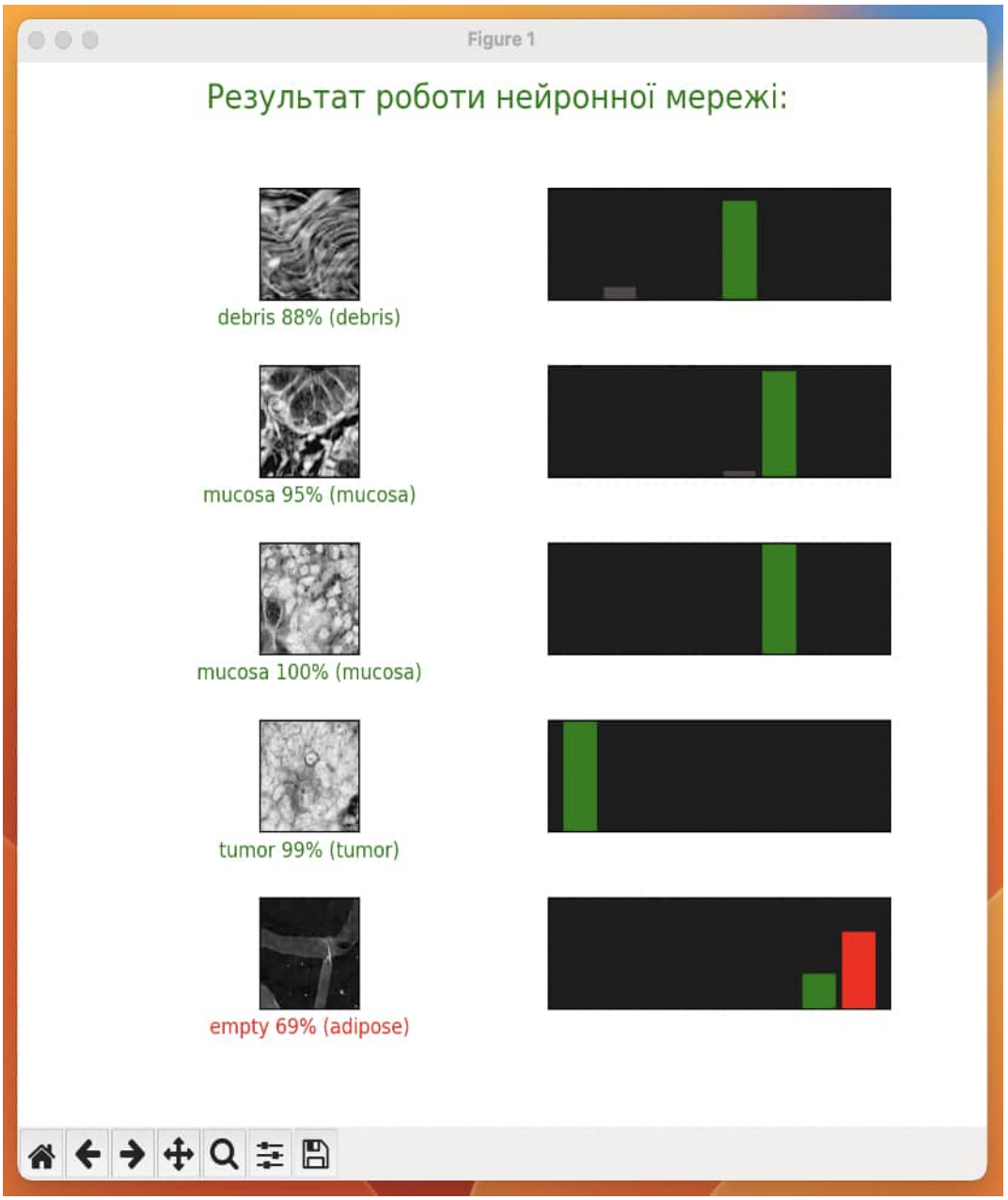

Рис. 2.14 Результат виконання

ВИСНОВОК

Під час виконання дипломної роботи я ознайомився зі згортковими нейронними мережами, з основною структурою CNN, її шарами та операціями, які вона виконує, а також з загальним виглядом і алгоритмами обробки з використанням фільтрів.

Було реалізовано нейронну мережу, яка класифікує зображення в гістології колоректального раку. Гістологія колоректального раку - це галузь медицини, яка вивчає структуру та склад тканин та клітин, які є характерними для раку товстої кишки та прямої кишки. Це важливо, оскільки колоректальний рак є одним з найпоширеніших видів раку у світі. Раннє виявлення раку та правильне лікування може значно підвищити шанси на успішне вилікування та збереження життя пацієнта.

Машинне навчання та штучний інтелект є перспективними технологіями для майбутнього медицини. Штучний інтелект, який є найбільш продуктивною складовою частиною цих технологій, дозволяє комп'ютерам аналізувати та обробляти велику кількість медичних даних. Це може допомогти лікарям в прийнятті рішень на основі високоякісної інформації та максимально точної діагностики.

Посилання та література

1. Нейронні мережі - шлях до глибинного навчання. Codeguida. URL: https://codeguida.com/post/739 .

2. CNN | Introduction to Pooling Layer - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

3. Convolutional Neural Networks' mathematics. URL: https://towardsdatascience.com/convolutional-neural-networks-mathematics-1beb3e6447c0?gi=70342d1660a9

4. What are convolutional neural networks (CNN)?. TechTalks. URL: https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/

5. Contributors to Wikimedia projects. Convolutional neural network - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network

6. What are Convolutional Neural Networks? | IBM. IBM - Deutschland | IBM. URL: https://www.ibm.com/topics/convolutional-neural-networks (date of access: 17.05.2023).