СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи

_____магістр_____

(освітньо-кваліфікаційний рівень)

на тему ___Розпізнавання зображень на основі штучних нейронних мереж для ідентифікації сцен антисоціального характеру_____

Виконав: студент _2_курсу, групи_ІСТ-21дм_

__ ___126 «Інформаційні системи та технології__

(шифр і назва спеціальності) .

_____Богомолов А.С._____

(прізвище та ініціали) .

Керівник:_____Лифар В.О._____

(прізвище та ініціали) .

Рецензент_____Кряжич О.О._____

(прізвище та ініціали)

Сєвєродонецьк – 2022 року

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ ДО МАГІСТЕРСЬКОЇ ДИПЛОМНОЇ РОБОТИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Освітньо-кваліфікаційний рівень _____магістр_____

Спеціальність _____126 «Інформаційні системи та технології__

(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ .

Завідувач кафедри ІТП .

_____д.т.н., доц. Лифар В.О.

(підпис) .

«____»_____2022 р. .

ЗАВДАННЯ

на магістерську дипломну роботу студенту

_____Богомолову Андрію Сергійовичу_____

(прізвище, ім’я, по батькові)

1. Тема роботи _Розпізнавання зображень на основі штучних нейронних мереж для ідентифікації сцен антисоціального характеру_____

керівник роботи _____Лифар Володимир Олексійович, проф, д.т.н _____

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «___»____2022_року №___

2. Строк подання студентом роботи_____

3. Вихідні дані до роботи_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

_____

_____

5. Перелік графічного матеріалу (з точним значенням обов’язків креслень)_____

_____

_____

6. Консультанти розділів проекту (роботи)

Розділ Прізвище, ініціали та посада

консультанта

Підпис, дата

завдання видав завдання

прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів дипломної роботи	Строк виконання етапів роботи
1	Консультація та затвердження тематики	06.09.2022 – 13.09.2022
2	Аналіз та дослідження обранної тематики	13.09.2022 – 14.10.2022
3	Аназіз сформульованої теми на основі вибранної тематики	16.10.2022 – 24.10.2022
4	Розробка алгоритму розробки штучної нейронної мережі	24.10.2022 – 05.11.2022
5	Укладання, оформлення та погодження пояснювальної записки з керівником	05.11.2022 – 21.11.2022

Студент _____Богомолов А.С._____

(прізвище та ініціали) (підпис)

Керівник роботи _____Лифар В.О._____

(прізвище та ініціали) (підпис)

РЕФЕРАТ

Робота містить 49 сторінок основного тексту, 7 сторінок додатків, 31 малюнок, використаних 15 джерел.

У ході виконання даної дипломної роботи були проведені дослідження предметної області навчання нейронної мережі за допомогою методу згортуючих нейронни мереж.

Метою роботи є розробка нейронної мережі, її навчання за заданими категоріями, та перевірка навченної моделі нейронної мережі із подальшим використанням для модерації завантажуваних зображень на Discord сервер.

Було проведено дослідження методів налаштування моделі нейроннох мережі та засобів максимізації більш точних вихідних вихідних даних начання.

Ключові слова: НЕЙРОННА МЕРЕЖА, МОДЕЛЬ, ЗОБРАЖЕННЯ АНТИСОЦІАЛЬНОГО ХАРАКТЕРУ, НАВЧАННЯ, ТОЧНІСТЬ, ВТРАТИ, TENSORFLOW, KERAS

ABSTRACT

The work contains 49 pages of the main text, 7 pages of appendices, 31 pictures, 15 sources used.

In the course of this thesis, research was conducted on the subject area of neural network learning using the method of convolutional neural networks.

The purpose of the work is the development of a neural network, its training according to given categories, and verification of the trained model of the neural network with subsequent use for the moderation of images uploaded to the Discord server.

Research has been conducted on neural network model tuning methods and means of maximizing more accurate initial starting data.

Keywords: NEURAL NETWORK, MODEL, IMAGE OF ANTISOCIAL CHARACTER, LEARNING, ACCURACY, LOSS, TENSORFLOW, KERAS

Зміст

ВСТУП

Актуальність досліджень. У сьогоденному житті актуальність нейронніх мереж обусловлена тенденцією BigData, у той самий час, коли компанії накопичують чималі масиви із даними, і паралельні обчислення дали вченим дані для навчання та обчислювальні ресурси, необхідні для роботи заплутаних штучних нейронних мереж. Нейронні мережі вирішують проблеми, які потребують розпізнавання образів. Наприклад, нейронну мережу можна навчити розпізнавати рукописні цифри. Іншим прикладом є безпілотний автомобіль Google, який навчений класично розпізнавати собаку, вантажівку чи автомобіль. Вони підходять для розпізнавання образів, класифікації та оптимізації. Це включає розпізнавання рукописного тексту, розпізнавання обличчя, розпізнавання мови, переклад тексту, виявлення шахрайства з кредитними картками, медичну діагностику та рішення для величезних обсягів даних. Його можна використовувати для пошуку зв’язків між шаблонами, для перетворення одного типу даних в інший і створення асоціацій або узагальнень між різними сутностями.

Об'єкт досліджень: Розпізнавання зображень антисоціального характеру

Предмет досліджень: Штучні нейронні мережі

Мета дослідження: Навчити штучну нейронну мережу розпізнавати зображення агтисоціального характеру під час загруски на сервер

Завдання дослідження: Розробка штучної нейронної мережі

1. РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ НА ОСНОВІ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

1.1. Штучні нейронні мережі загальний опис

Нейронні мережі, так як штучні нейронні мережі або імітовані нейронні мережі, являють собою підгрупу машинного навчання та є основою алгоритмів глибокого навчання. Така назва і структура навіяні людським мозком, імітуючи спосіб, яким біологічні нейрони передають сигнали один одному.

Штучні нейронні мережі створюються з вузлових шарів, які мають у собі початковий рівень, декілька прихованих шарів та вихідний рівень. Штучні нейрони, пов’язуються між собою та мають відповідні ваги та пороги. Якщо вихід якогось із вузлів переповлює зазначений поріг, то цей вузол активується, відправляючи дані далі на наступний рівень мережі. Інакше дані залишаються.

Нейронні мережі використовують початкові дані, для навчання та підвищення своєї точності з часом. При налаштуванні аглоритмів для навчання на точність, тоді можна отримати потужні інструменти в інформаційному просторі та штучному інтелекті, що дозволяє обробляти дані з високою швидкістю. Розпізнавання мовлення або розпізнавання зображень може займати години порівняно з ідентифікацією експертів-людей.

У світі фінансів можна використовувати нейронні мережі для прогнозування часових рядів, класифікація цінних паперів, моделювання кредитного ризику та побудова власних породжаючих цін.

Нейронна мережа працює за схожим алгоритмом до нейронної мережі людського мозку. «Нейрон» — це математична функція, яка збирає та класифікує інформацію відповідно до певної архітектури.

1.1.1. Як працюють нейронні мережі?

Архітектура нейронних мереж повторює структуру людського мозку. Клітини людського мозку, звані нейронами, утворюють складну мережу з високим ступенем взаємозв'язку та посилають один одному електричні сигнали, допомагаючи людям обробляти інформацію. Так само штучна нейронна мережа складається з штучних нейронів, які взаємодіють на вирішення проблем. Штучні нейрони - це програмні модулі, які називаються вузлами, а штучні нейронні мережі - це програми або алгоритми, які використовують обчислювальні системи для виконання математичних обчислень.

1.1.2. Архітектура базової нейронної мережі

Базова нейронна мережа містить три шари взаємопов'язаних штучних нейронів:

- **Вхідний шар**

Інформація із зовнішнього світу надходить у штучну нейронну мережу із вхідного шару. Вхідні вузли обробляють дані, аналізують або класифікують їх та передають на наступний шар.

- **Прихований шар**

Приховані шари одержують вхідні дані від вхідного шару або інших прихованих шарів. Штучні нейронні мережі можуть мати велику кількість прихованих шарів. Кожен прихований шар аналізує вихідні дані попереднього шару, обробляє їх та передає на наступний шар.

- **Вихідний шар**

Вихідний шар дає остаточний результат обробки всіх даних штучною нейронною мережею. Він може мати один чи кілька вузлів. Наприклад, під час вирішення завдання двійкової класифікації (так/ні) вихідний шар матиме один вихідний вузол, який дасть результат «1» чи «0». Однак у разі множинної класифікації вихідний шар може складатися більш ніж з одного вихідного вузла.

1.1.3. Архітектура глибокої нейронної мережі

Глибокі нейронні мережі або мережі глибокого навчання мають кілька прихованих шарів із мільйонами зв'язаних один з одним штучних нейронів. Число, зване вагою, свідчить про зв'язку одного вузла коїться з іншими. Вага є позитивним числом, якщо один вузол збуджує інший або негативним, якщо один вузол пригнічує інший. Вузли з вищими значеннями ваги мають більший вплив інші вузли.

Теоретично глибокі нейронні мережі можуть зіставляти будь-який тип введення будь-яким типом виведення. Однак варто враховувати, що їм потрібне набагато складніше навчання, ніж інші методи машинного навчання. Таким вузлам потрібні мільйони прикладів навчальних даних, а не сотні чи тисячі, як у випадку із простими мережами.

1.1.4. Які типи нейронних мереж є?

Штучні нейронні мережі можна класифікувати після того, як дані передаються від вхідного вузла до вихідного вузла. Нижче наведено кілька прикладів.

Нейронні мережі прямого розповсюдження

Нейронні мережі прямого поширення обробляють дані в одному напрямку від вхідного вузла до вихідного вузла. Кожен вузол одного шару пов'язаний із кожним вузлом наступного шару. Нейронні мережі прямого

поширення використовують процес зворотний зв'язок поліпшення прогнозів з часом.

Алгоритм зворотного розповсюдження

Штучні нейронні мережі постійно навчаються, використовуючи цикли зворотного зв'язку, що коригують, для поліпшення своєї прогностичної аналітики. Простіше кажучи, йдеться про дані, що протікають від вхідного вузла до вихідного вузла по безлічі різних шляхів нейронної мережі. Правильним є лише один шлях, який зіставляє вхідний вузол із правильним вихідним вузлом. Щоб знайти цей шлях, нейронна мережа використовує петлю зворотного зв'язку, яка працює так:

1. Кожен вузол робить припущення про наступний вузол на дорозі.

2. Він перевіряє, чи є припущення правильним. Вузли надають більш високі значення ваги коліям, які призводять до більш правильних припущень, і нижчі значення ваги колій вузлів, які призводять до неправильних припущень.

3. Для наступної точки даних вузли роблять новий прогноз, використовуючи шляхи з вищою вагою, а потім повторюють крок 1.

Згорткові нейронні мережі

Приховані шари в нейронних мережах згортання виконують певні математичні функції (наприклад, підсумовування або фільтрацію), звані згортками. Вони дуже корисні для класифікації зображень, оскільки можуть витягувати з них відповідні ознаки, корисні для розпізнавання та класифікації. Нову форму легше обробляти без втрати функцій, які мають вирішальне значення для правильного припущення. Кожен прихований шар витягує та обробляє різні характеристики зображення: межі, колір та глибину.

1.1.5. Що таке глибоке навчання у контексті нейронних мереж?

Штучний інтелект - це область комп'ютерних наук, яка досліджує методи надання машинам можливості виконувати завдання, які потребують людського інтелекту. Машинне навчання - це метод штучного інтелекту, який дає комп'ютерам доступ до дуже великих наборів даних для подальшого навчання. Програмне забезпечення для машинного навчання знаходить шаблони у існуючих даних та застосовує ці шаблони до нових даних для прийняття розумних рішень. Глибоке навчання - це різновид машинного навчання, в якому для обробки даних використовуються мережі глибокого навчання.

1.1.6. Машинне навчання та глибоке навчання

Традиційні методи машинного навчання вимагають участі людини, щоб програмне забезпечення працювало належним чином. Фахівець із роботи з даними вручну визначає набір відповідних функцій, які має аналізувати програмне забезпечення. Це обмеження робить створення та управління програмним забезпеченням стомлюючим та трудомістким процесом.

З іншого боку, при глибокому навчанні фахівець із роботи з даними надає програмному забезпеченню лише необроблені дані. Мережа глибокого навчання здобуває функції самостійно та навчається більш незалежно. Вона може аналізувати неструктуровані набори даних (наприклад, текстові документи), визначати пріоритети атрибутів даних та вирішувати складніші завдання.

Наприклад, під час навчання програмного забезпечення з алгоритмами машинного навчання правильно ідентифікувати зображення тварини вам потрібно виконати такі кроки:

Знайти і вручну відзначити тисячі зображень свійських тварин: кішок, собак, коней, хом'яків, папуг і т.д.

Повідомити програмне забезпечення з алгоритмами машинного навчання, які функції необхідно знайти, щоб воно могло ідентифікувати зображення методом виключення. Наприклад, воно може підрахувати кількість ніг, а потім перевірити форму очей, вух, хвоста, колір хутра тощо.

Вручну оцінити та змінити позначені набори даних, щоб підвищити точність програмного забезпечення. Наприклад, якщо у вашому тренувальному наборі занадто багато зображень чорних кішок, програмне забезпечення правильно визначить чорну кішку, але не білу.

При глибокому навчанні нейронні мережі будуть обробляти всі зображення та автоматично визначати, що спочатку їм потрібно проаналізувати кількість ніг та форму морди, а вже потім подивитися на хвости, щоб правильно ідентифікувати тварину на зображенні.

1.2. Що таке зображення?

Перш ніж перейти до обробки зображень, нам потрібно спочатку зрозуміти, що саме являє собою зображення. Зображення представлено своїми розмірами (висотою та шириною) на основі кількості пікселів. Наприклад, якщо розміри зображення 500 x 400 (ширина x висота), загальна кількість пікселів у зображенні становить 200 000.

Цей піксель — це точка на зображенні, яка набуває певного відтінку, непрозорості або кольору. Зазвичай він представлений в одному з наступного:

Відтінки сірого – піксель – це ціле число зі значенням від 0 до 255 (0 – повністю чорний, а 255 – повністю білий).

RGB – піксель складається з 3 цілих чисел від 0 до 255 (цілі числа представляють інтенсивність червоного, зеленого та синього кольорів).

RGBA – це розширення RGB з доданим полем альфа, яке представляє непрозорість зображення.

Обробка зображення вимагає фіксованої послідовності операцій, які виконуються з кожним пікселем зображення. Процесор зображень виконує першу послідовність операцій із зображенням піксель за пікселем. Коли це буде повністю зроблено, він почне виконувати другу операцію тощо. Вихідне значення цих операцій можна обчислити для будь-якого пікселя зображення.

1.2.1. Що таке обробка зображень?

Обробка зображення – це процес перетворення зображення в цифрову форму та виконання певних операцій для отримання з нього корисної інформації. Система обробки зображень зазвичай обробляє всі зображення як 2D-сигнали, застосовуючи певні заздалегідь визначені методи обробки сигналів.

Існує п'ять основних типів обробки зображень:

Візуалізація - знайдіть об'єкти, які не видно на зображенні

Розпізнавання - розрізнення або виявлення об'єктів на зображенні

Збільшення різкості та відновлення — створіть покращене зображення з оригінального зображення

Розпізнавання візерунків. Вимірюйте різноманітні візерунки навколо об’єктів на зображенні

Отримання - перегляд і пошук зображень у великій базі даних цифрових зображень, схожих на вихідне зображення

1.2.2. Основні етапи обробки зображень

Отримання зображень

Отримання зображення є першим кроком в обробці зображення. Цей крок також відомий як попередня обробка в обробці зображень. Він передбачає отримання зображення з джерела, зазвичай апаратного джерела.

Поліпшення зображення

Покращення зображення – це процес виявлення та виділення певних цікавих особливостей зображення, яке було затемненим. Це може включати зміну яскравості, контрастності тощо.

Відновлення зображення

Реставрація зображення – це процес покращення зовнішнього вигляду зображення. Однак, на відміну від покращення зображення, відновлення зображення виконується за допомогою певних математичних або імовірнісних моделей.

Обробка кольорових зображень

Обробка кольорових зображень включає низку методів моделювання кольорів у цифровій сфері. Цей крок набув популярності завдяки значному використанню цифрових зображень в Інтернеті.

1.2.3. Вейвлети та обробка з різною роздільною здатністю

Вейвлети використовуються для представлення зображень із різною роздільною здатністю. Зображення поділяються на вейвлети або менші області для стиснення даних і пірамідального представлення.

Компресія

Стиснення – це процес, який використовується для зменшення обсягу пам’яті, необхідного для збереження зображення, або пропускної здатності, необхідної для його передачі. Особливо це робиться, коли зображення призначене для використання в Інтернеті.

Морфологічна обробка

Морфологічна обробка — це набір операцій обробки для морфінгу зображень на основі їх форм.

Сегментація

Сегментація є одним із найскладніших етапів обробки зображень. Це передбачає поділ зображення на складові частини або об’єкти.

Зображення та опис

Після того, як зображення сегментується на області в процесі сегментації, кожна область представляється та описується у формі, придатній для подальшої комп'ютерної обробки. Репрезентація має справу з характеристиками образу та регіональними властивостями. Опис займається вилученням кількісної інформації, яка допомагає відрізнити один клас об'єктів від іншого.

Визнання

Розпізнавання присвоює мітку об'єкту на основі його опису.

1.2.4. Застосування обробки зображень

Отримання медичних зображень

Обробка зображень широко використовується в медичних дослідженнях і дозволяє складати більш ефективні та точні плани лікування. Наприклад, його можна використовувати для раннього виявлення раку молочної залози за допомогою складного алгоритму виявлення вузликів під час сканування молочної залози. Оскільки медичне використання потребує висококваліфікованих процесорів зображень, ці програми потребують значного впровадження та оцінки, перш ніж їх можна буде прийняти до використання.

Технології визначення дорожнього руху

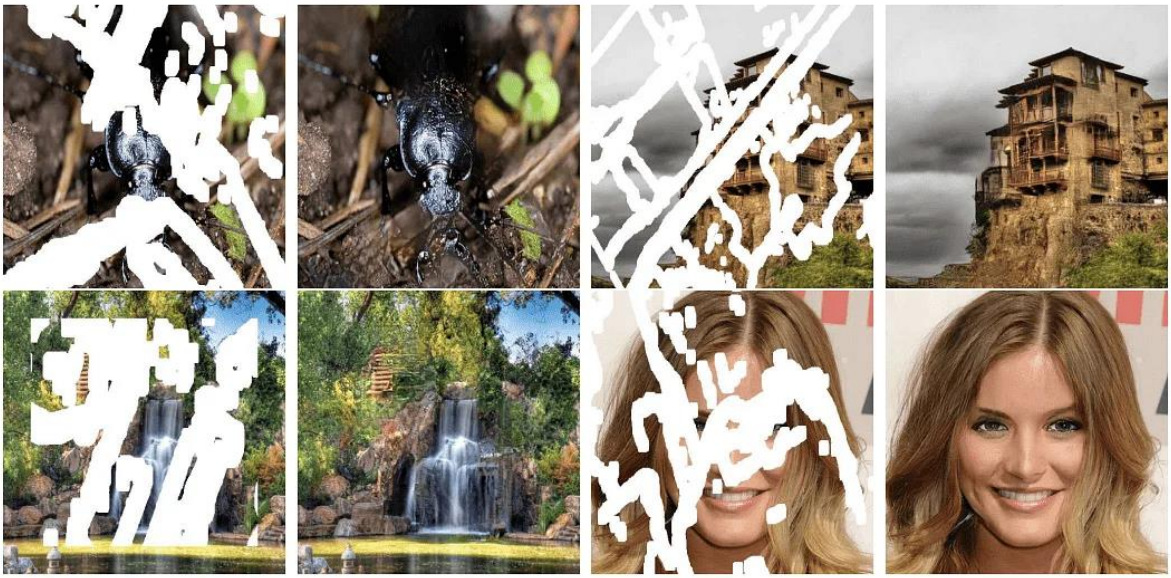
У випадку датчиків дорожнього руху ми використовуємо систему обробки відеозображення або VIPS. Він складається з а) системи захоплення зображення, б) телекомунікаційної системи та в) системи обробки зображень. Під час зйомки відео VIPS має кілька зон виявлення, які видають сигнал «увімкнено», коли транспортний засіб в'їжджає в зону, а потім видають сигнал «вимкнено», коли транспортний засіб виходить із зони виявлення. Ці зони виявлення можуть бути налаштовані для кількох смуг і використовуватися для виявлення руху на конкретній станції.

Малюнок 1 Обробка зображення дорожноїго руху

Крім того, він може автоматично записувати номерний знак транспортного засобу, розрізняти тип транспортного засобу, стежити за швидкістю водія на трасі та багато іншого.

Реконструкція зображення

Обробку зображення можна використовувати для відновлення та заповнення відсутніх або пошкоджених частин зображення. Це передбачає використання систем обробки зображень, які пройшли інтенсивне навчання з наявними наборами даних фотографій для створення нових версій старих і пошкоджених фотографій.



Малюнок 2 Відновлення пошкоджених зображень

Розпізнавання обличчя

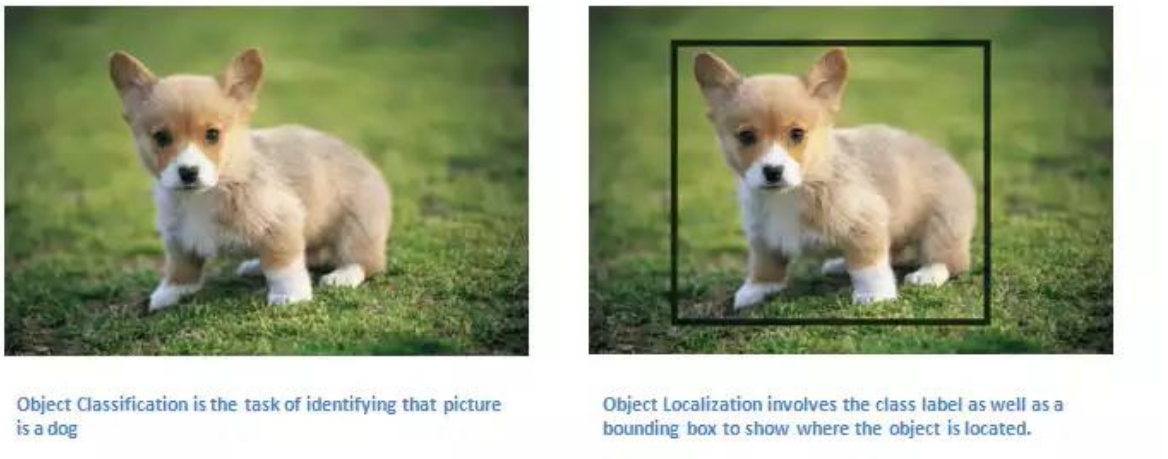
Одним із найпоширеніших застосувань обробки зображень, які ми використовуємо сьогодні, є розпізнавання облич. Він дотримується алгоритмів глибокого навчання, коли машина спочатку навчається конкретним характеристикам людських облич, таким як форма обличчя, відстань між очима тощо. Після навчання машини цим рисам людського обличчя вона почне приймати всі предмети на зображенні, які нагадують людське обличчя. Розпізнавання обличчя є життєво важливим інструментом, який використовується для безпеки, біометрії та навіть фільтрів, доступних сьогодні в більшості програм соціальних мереж.

1.3. Що таке класифікація зображень і комп’ютерний зір?

Класифікація зображень - це завдання упізнання зображень і класифікації їх в одному з кількох попередньо визначених окремих класів. Таким чином, ПЗ та програми для впізнання зображень можуть визначити, які об'єкти зображені на зображенні та відрізняти об'єкти один від одного.

Галузь дослідження, спрямована для забезпечення машин даною здатністю, називають комп’ютерним зором. Класифікація зображень є основою для вирішення численні проблеми CV, будучи одним із завдань комп’ютерного зору (CV), включаючи:

Класифікація зображень із місцезнаходженням – розташування зображення в заданому класі та малювання рамки навколо шукаємого об’єкта, для того щоб можна було показати, де цей об'єкт розташований на зображенні.

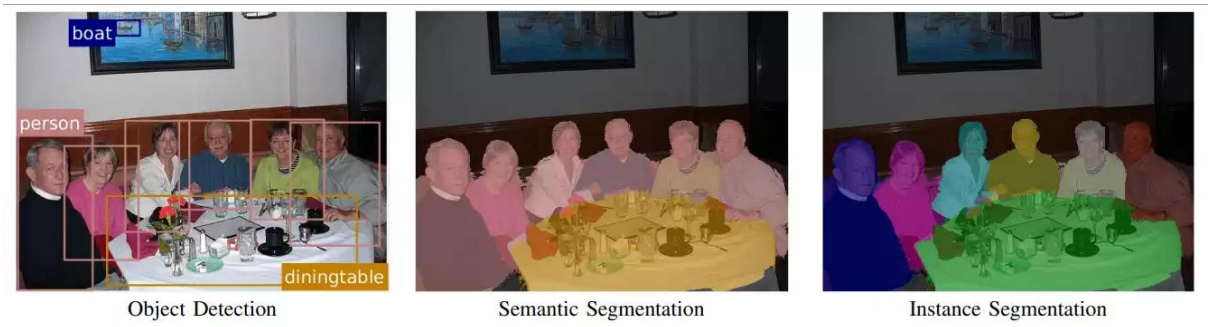


Малюнок 3 Класифікація зображень проти класифікації зображень із локалізацією

Знаходження об’єктів – класифікація кількох різних об’єктів на зображенні і відображення розташування їх на зображенні за допомогою обмежувальних рамок. Значить, це вид класифікації зображень із завданнями місцезнаходження для багатьох об’єктів.

Семантична сегментація – ідентифікація специфічних пікселів, які відноситься для усіх об’єкта на зображенні, замість малювання рамок навколо усіх об’єкта, як при знаходженні об’єкта.

Сегментація екземплярів – диференціація кількох екземплярів, що належать до одного класу.



Малюнок 4 Різниця між виявленням об’єктів, семантичною сегментацією та сегментацією екземплярів.

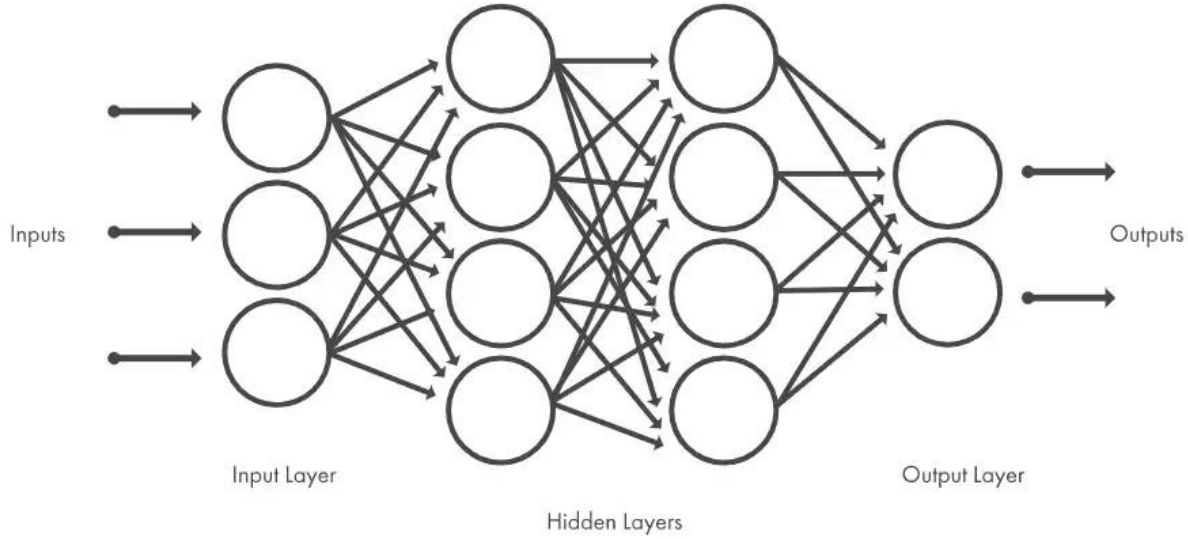
Дослідники можуть використовувати моделі глибокого навчання для вирішення завдань комп’ютерного зору. Глибоке навчання – це

спрямована техніка машинного навчання на навчання машин навчатися на прикладі. Оскільки переварно методи глибокого навчання використовують архітектури нейронних мереж, тому моделі для глибокого навчання також можна називати глибокими нейронними мережами.

1.3.1. Глибокі нейронні мережі: «як» розпізнавання зображень та інші методи комп'ютерного зору

Розпізнавання зображень є одним із завдань, у яких чудово працюють глибокі нейронні мережі (DNN). Нейронні мережі — це обчислювальні системи, використовувані для визначення шаблонів. Їхня архітектура натхненна структурою людського мозку, тому так називаються. Вони складаються з трьох типів шарів: вхідні, приховані шари та вихідні. Вхідний рівень отримує сигнал, прихований рівень обробляє його, а вихідний рівень приймає рішення або прогноз щодо вхідних даних. Кожен рівень мережі складається з штучних нейронів, які виконують обчислення.

Коли можна назвати нейронну мережу глибокою? Коли нейронна мережа містить сотні прихованих шарів, у той час як традиційні нейронні мережі мають до трьох прихованих шарів.



Малюнок 5 В архітектурі нейронної мережі кожен рівень складається з вузлів. Кількість прихованих шарів необов’язкова.

1.3.2. Як нейронні мережі навчаються розпізнавати шаблони

Як зрозуміти, хто проходить на вулиці знайомий чи незнайомий (таких ускладнень, як короткозорість без урахування)? Ми дивимося на них, підсвідомо аналізуємо їхню зовнішність і якщо невід'ємне особливості — форма обличчя, колір очей, зачіска, тип статури, хода — збігаються з людиною, яку ми знаємо, ми впізнаємо її.

Для щоб система могла розпізнавати обличчя, вона повинна спочатку вивчити їхні особливості. Її потрібно навчити відрізняти об’єкт X від об’єкта Y. Моделі глибокого навчання вивчають ці характеристики поіншому, ніж моделі машинного навчання (ML). Саме тому підходи до модельного навчання різні.

1.3.3. Навчання моделей глибокого навчання (таких як нейронні мережі)

Щоб побудувати модель ML, яка може, наприклад, передбачити відтік клієнтів, дослідники даних повинні вказати, які вхідні характеристики (властивості проблеми) модель враховуватиме при

прогнозуванні результату. Це може бути освіта клієнта, дохід, етап життєвого циклу, функції продукту або використовувані модулі, кількість взаємодій зі службою підтримки та їх результати. Процес побудови функцій із використанням знань предметної області називається розробкою функцій.

Якби ми навчили модель глибокого навчання, щоб побачити різницю між собакою та котом за допомогою розробки функцій… Ну, уявіть собі, що ми збираємо характеристики мільярдів котів і собак, які живуть на цій планеті. Ми не можемо побудувати точні характеристики, які працюватимуть для кожного можливого зображення, враховуючи такі ускладнення, як мінливість об’єкта, що залежить від точки огляду, безлад фону, умови освітлення або деформація зображення. Має бути інший підхід, і він існує завдяки природі нейронних мереж.

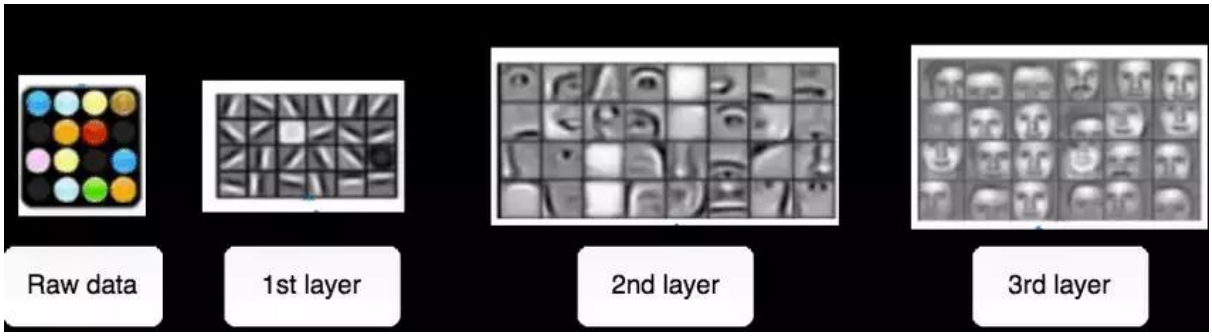
Нейронні мережі вивчають функції безпосередньо з даних, з якими вони навчаються, тому фахівцям не потрібно витягувати функції вручну.

«Потужність нейронних мереж полягає в їхній здатності вивчати представлення ваших навчальних даних і те, як найкраще пов’язати їх із вихідною змінною, яку ви хочете передбачити. У цьому сенсі нейронні мережі вивчають відображення. З математичної точки зору вони здатні вивчати будь-яку функцію відображення, і було доведено, що вони є універсальними алгоритмами наближення», – зазначає Джейсон Браунлі в прискореному курсі багаторівневих нейронних мереж персептронів.

Навчальні дані, у цьому випадку, є великим набором даних, який містить багато прикладів кожного класу зображень. Коли ми говоримо про великий набір даних, ми дійсно це маємо на увазі. Наприклад, набір даних ImageNet містить понад 14 мільйонів зображень, анотованих людиною, що представляють 21 841 поняття (набори синонімів або синсети відповідно до ієрархії WordNet), у середньому 1000 зображень на поняття.

Кожне зображення анотовано (позначено) категорією, до якої воно належить – кішка чи собака. Алгоритм досліджує ці приклади, дізнається про візуальні характеристики кожної категорії та, зрештою, дізнається, як розпізнавати кожен клас зображення. Цей модельний стиль навчання називається навчанням під наглядом.

Кожен шар вузлів тренується на виході (наборі функцій), створеному попереднім шаром. Таким чином, вузли в кожному наступному шарі можуть розпізнавати більш складні, детальні характеристики – візуальні представлення того, що зображено на зображенні. Така «ієрархія зростаючої складності та абстракції» відома як ієрархія функцій.



Малюнок 6 Приклад ієрархії ознак, отриманої за допомогою моделі глибокого навчання на обличчях.

Отже, чим більше шарів має мережа, тим більша її здатність передбачити.

Провідною архітектурою, яка використовується для задач розпізнавання та виявлення зображень, є згорткові нейронні мережі (CNN). Згорткові нейронні мережі складаються з декількох шарів з маленькими колекціями нейронів, кожен з яких сприймає невеликі частини зображення. Результати всіх колекцій у шарі частково перекриваються таким чином, щоб створити повне представлення зображення. Потім шар нижче повторює цей процес на новому

представленні зображення, дозволяючи системі дізнатися про композицію зображення.

Історія глибоких CNN сягає початку 1980-х років. Але тільки в 2010-х роках дослідникам вдалося досягти високої точності вирішення завдань розпізнавання зображень за допомогою глибоких згорткових нейронних мереж. як? Вони почали навчати та розгортати CNN, використовуючи графічні процесори (GPU), які значно прискорюють складні системи на основі нейронних мереж. Обсяг навчальних даних – фотографій чи відео – також збільшився, оскільки камери мобільних телефонів і цифрові камери почали швидко розвиватися та стали доступними.

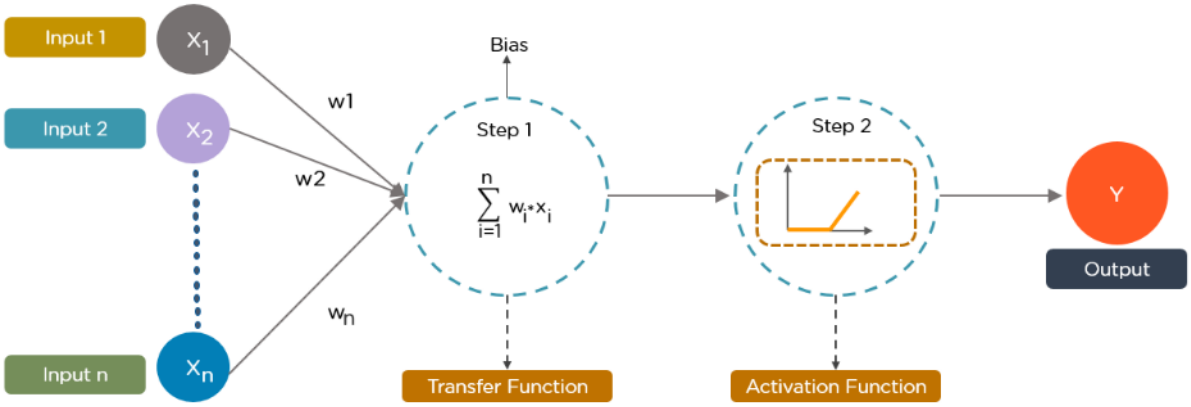
2. РОЗДІЛ 2. МЕТОДИ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Штучний інтелект пройшов довгий шлях і плавно подолав розрив між потенціалом людей і машин. І ентузіасти даних у всьому світі працюють над численними аспектами штучного інтелекту та втілюють бачення в реальність, і однією з таких дивовижних сфер є комп'ютерне бачення.

2.1. Визначення нейронних мереж

Нейронна мережа структурована як людський мозок і складається зі штучних нейронів, також відомих як вузли. Ці вузли розташовані поруч один з одним у три шари:

- Вхідний шар
- Прихований шар(и)
- Вихідний шар



Малюнок 7 Проста схема роботи нейронної мережі

Дані надають кожному вузлу інформацію у формі вхідних даних. Вузол множить вхідні дані на випадкові ваги, обчислює їх і додає зміщення. Нарешті, нелінійні функції, також відомі як функції активації, застосовуються для визначення того, який нейрон запускати.

2.2. Як працюють алгоритми глибокого навчання?

Хоча алгоритми глибокого навчання мають представлення самонавчання, вони залежать від ШНМ, які відображають те, як мозок обчислює інформацію. Під час процесу навчання алгоритми використовують невідомі елементи у розподілі вхідних даних, щоб витягувати функції, групувати об’єкти та виявляти корисні шаблони даних. Подібно до навчання машин для самонавчання, це відбувається на кількох рівнях із використанням алгоритмів для створення моделей.

Моделі глибокого навчання використовують кілька алгоритмів. Хоча жодна мережа не вважається ідеальною, деякі алгоритми краще підходять для виконання конкретних завдань. Щоб вибрати правильні алгоритми, добре отримати чітке розуміння всіх основних алгоритмів.

2.3. Типи алгоритмів, які використовуються в Deep Learning

2.3.1. Згорточні нейронні мережі (CNN)

CNN, також відомі як ConvNets, складаються з кількох рівнів і в основному використовуються для обробки зображень і виявлення об’єктів. Янн ЛеКун розробив перший CNN у 1988 році, коли він називався LeNet. Він використовувався для розпізнавання таких символів, як поштові індекси та цифри.

CNN широко використовуються для ідентифікації супутникових зображень, обробки медичних зображень, прогнозування часових рядів і виявлення аномалій.

Як працюють CNN?

CNN мають кілька рівнів, які обробляють і витягують функції з даних:

Шар згортки

CNN має шар згортки, який має кілька фільтрів для виконання операції згортки.

Випрямлений лінійний блок (ReLU)

CNN мають рівень ReLU для виконання операцій над елементами. Результатом є виправлена карта ознак.

Рівень об'єднання

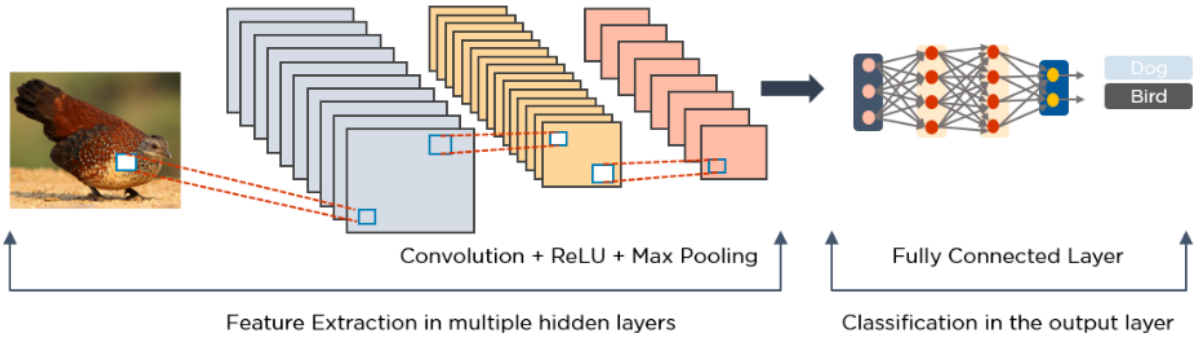
Далі виправлена карта функцій надходить у шар об’єднання. **Об’єднання** — це операція зменшення вибірки, яка зменшує розміри карти функцій.

Потім шар об’єднання перетворює отримані двовимірні масиви з об’єднаної карти об’єктів в єдиний довгий безперервний лінійний вектор шляхом його зведення.

Повністю підключений рівень

Повністю пов’язаний шар утворюється, коли сплощена матриця з шару об’єднання подається як вхідний сигнал, який класифікує та ідентифікує зображення.

Нижче наведено приклад зображення, обробленого через CNN.



Малюнок 8 Схема роботи методу CNN

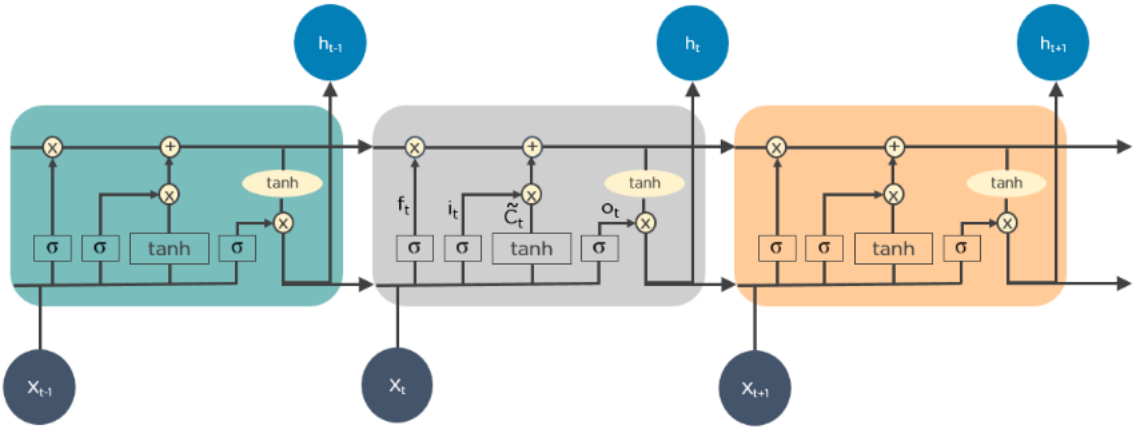
2.3.2. Мережі довготривалої короткострокової пам’яті (LSTM)

LSTM — це тип рекурентної нейронної мережі (RNN), яка може вивчати та запам’ятовувати довгострокові залежності. Згадування минулої інформації протягом тривалого часу є типовою поведінкою.

LSTM зберігають інформацію з часом. Вони корисні для прогнозування часових рядів, оскільки запам’ятовують попередні вхідні дані. LSTM мають ланцюгову структуру, де чотири взаємодіючі рівні взаємодіють унікальним способом. Окрім передбачення часових рядів, LSTM зазвичай використовуються для розпізнавання мови, композиції музики та розробки фармацевтичних препаратів.

Як працюють LSTM?

- По-перше, вони забувають несуттєві частини попереднього стану
- Далі вони вибірково оновлюють значення стану комірки
- Нарешті, вихід певних частин клітини стану



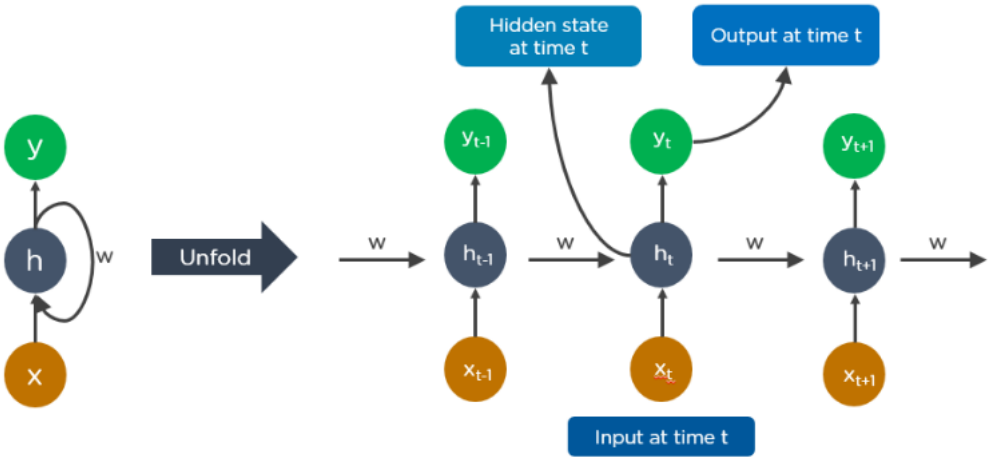
Малюнок 9 Схема роботи методу LSTM

2.3.3. Повторювані нейронні мережі (RNN)

RNN мають з’єднання, які утворюють спрямовані цикли, які дозволяють подавати вихідні сигнали з LSTM як входи до поточної фази.

Вихід із LSTM стає входом для поточної фази та може запам’ятовувати попередні входи завдяки своїй внутрішній пам’яті. RNN

зазвичай використовуються для підписів до зображень, аналізу часових рядів, обробки природної мови, розпізнавання рукописного тексту та машинного перекладу.



Малюнок 10 Схема роботи методу RNN

Як працюють RNN?

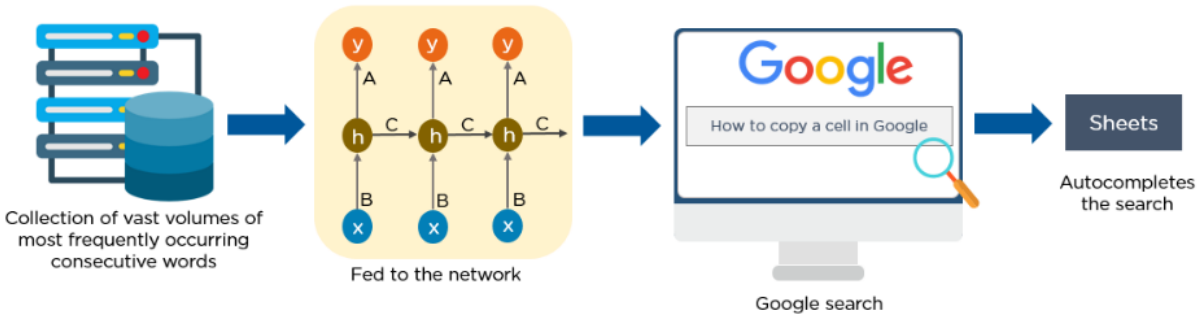
Вихід у момент часу t-1 подається на вхід у момент часу t.

Подібним чином, вихідний сигнал у момент часу t подається на вхід у момент часу t+1.

RNN можуть обробляти вхідні дані будь-якої довжини.

Обчислення враховує історичну інформацію, а розмір моделі не збільшується разом із розміром вхідних даних.

Ось приклад роботи функції автозавершення Google:



Малюнок 11 Приклад роботи RNN при пошуку в браузері

2.3.4. Генеративні змагальні мережі (GAN)

GAN — це генеративні алгоритми глибокого навчання, які створюють нові екземпляри даних, схожі на навчальні дані. GAN має два компоненти: генератор, який вчиться генерувати підроблені дані, і дискримінатор, який вчиться з цієї неправдивої інформації.

За певний період часу використання GAN зросло. Їх можна використовувати для покращення астрономічних зображень і моделювання гравітаційних лінз для дослідження темної матерії. Розробники відеоігор використовують GAN для покращення 2D-текстур із низькою роздільною здатністю у старих відеоіграх, відтворюючи їх у 4K або вищій роздільній здатності за допомогою навчання зображень.

GAN допомагають створювати реалістичні зображення та героїв мультфільмів, створювати фотографії людських облич і відтворювати 3D-об’єкти.

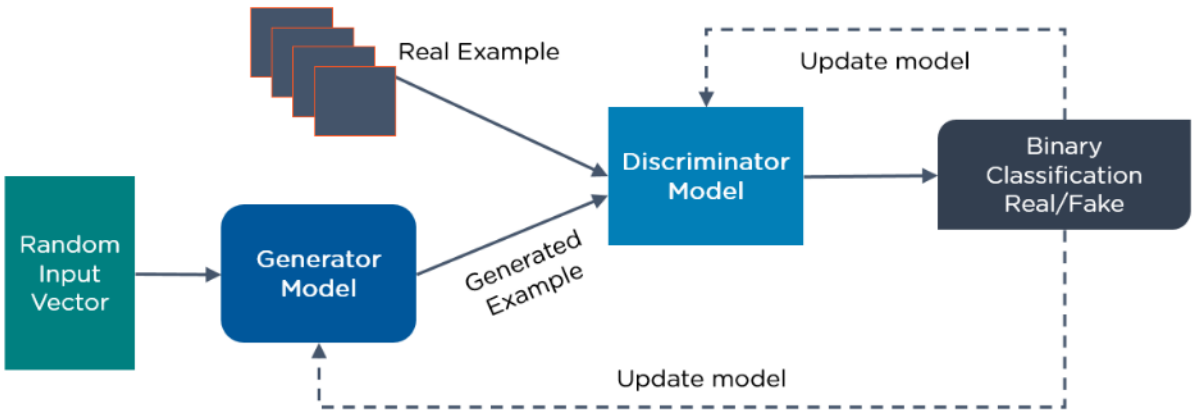
Як працюють GAN?

Дискримінатор вчиться розрізняти підроблені дані генератора від реальних даних зразка.

Під час початкового навчання генератор створює фальшиві дані, а дискримінатор швидко вчиться розпізнавати, що це хибність.

GAN надсилає результати генератору та дискримінатору для оновлення моделі.

Нижче наведено схему роботи GAN.



Малюнок 12 Схема роботи методу GAN

2.3.5. Радиально-базисні функціональні мережі (RBFN)

RBFN — це спеціальні типи прямої нейронної мережі, які використовують радіальні базисні функції як функції активації. Вони мають вхідний рівень, прихований рівень і вихідний рівень і переважно використовуються для класифікації, регресії та прогнозування часових рядів.

Як працюють RBFN?

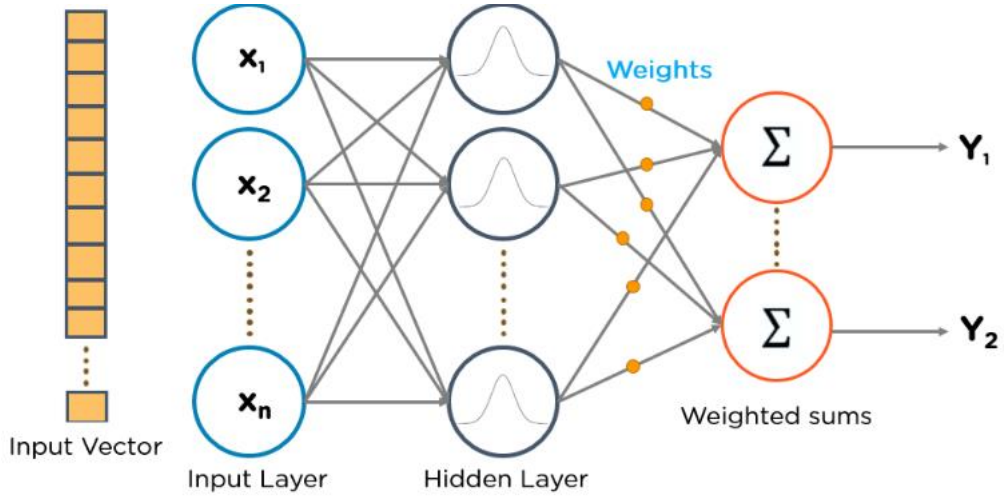
RBFN виконують класифікацію, вимірюючи схожість вхідних даних із прикладами з навчального набору.

RBFN мають вхідний вектор, який подається на вхідний рівень. Вони мають шар нейронів RBF.

Функція знаходить зважену суму вхідних даних, а вихідний рівень має один вузол на категорію або клас даних.

Нейрони в прихованому шарі містять функції передачі Гауса, які мають виходи, обернено пропорційні відстані від центру нейрона.

Вихід мережі є лінійною комбінацією радіально-базисних функцій входу та параметрів нейрона.



Малюнок 13 Схема роботи методу RBFN

2.3.6. Багатошарові персептрони (MLP)

MLPs є чудовим місцем для початку вивчення технології глибокого навчання.

MLP належать до класу прямої нейронної мережі з декількома шарами перцептронів, які мають функції активації. MLP складаються з вхідного рівня та вихідного рівня, які повністю з’єднані. Вони мають однакову кількість вхідних і вихідних рівнів, але можуть мати кілька прихованих рівнів і можуть використовуватися для створення програмного забезпечення для розпізнавання мовлення, розпізнавання зображень і машинного перекладу.

Як працюють MLP?

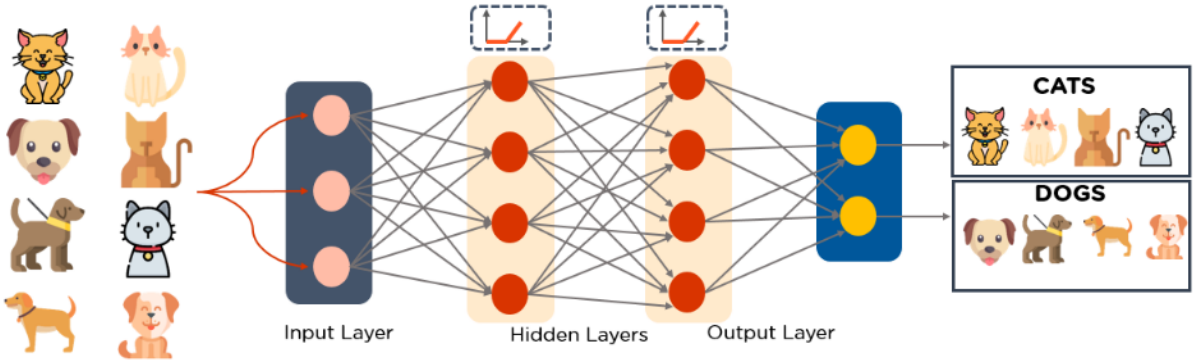
MLP передають дані на вхідний рівень мережі. Шари нейронів з'єднуються в граф, щоб сигнал проходив в одному напрямку.

MLP обчислюють вхідні дані з вагами, які існують між вхідним шаром і прихованими шарами.

MLP використовують функції активації, щоб визначити, які вузли запускати. Функції активації включають ReLU, сигмоподібні функції та tanh.

MLP тренують модель для розуміння кореляції та вивчення залежностей між незалежними та цільовими змінними з навчального набору даних.

Нижче наведено приклад MLP. Діаграма обчислює ваги та зміщення та застосовує відповідні функції активації для класифікації зображень котів і собак.



Малюнок 14 Схема роботи методу MLP

2.3.7. Самоорганізуючі карти (SOM)

Професор Теуво Кохонен винайшов SOM, які дозволяють візуалізувати дані, щоб зменшити розміри даних за допомогою самоорганізованих штучних нейронних мереж.

Візуалізація даних намагається вирішити проблему того, що люди не можуть легко візуалізувати багатовимірні дані. SOM створено, щоб допомогти користувачам зрозуміти цю багатовимірну інформацію.

Як працюють SOM?

SOM ініціалізують ваги для кожного вузла та вибирають вектор випадковим чином із навчальних даних.

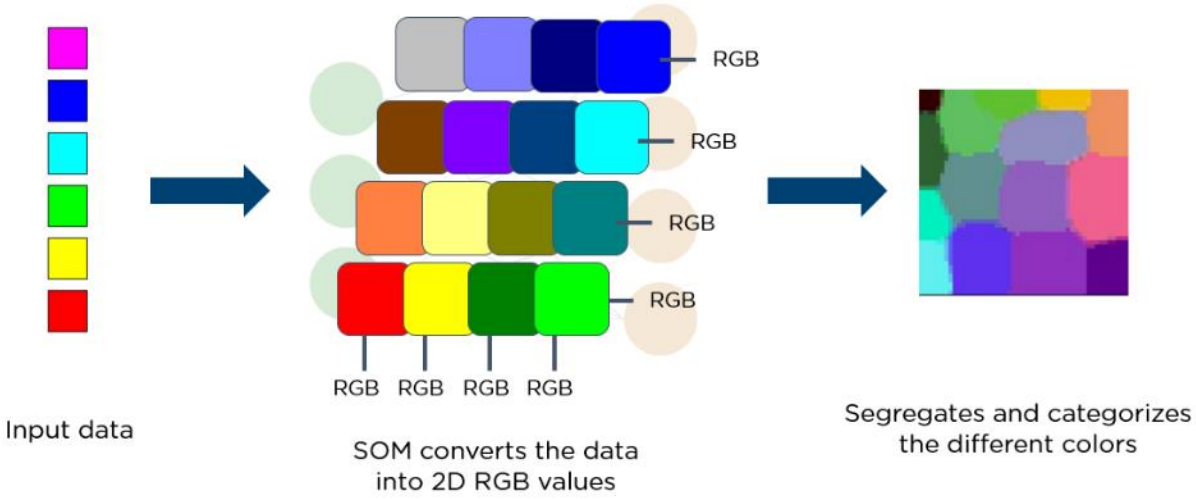
SOM досліджує кожен вузол, щоб визначити, які ваги є найбільш імовірним вхідним вектором. Вузол-переможець називається одиницею найкращого відповідності (BMU).

SOM виявляють околиці BMU, і з часом кількість сусідів зменшується.

SOM присуджує виграшну вагу вектору вибірки. Чим ближче вузол до BMU, тим більше змінюється його вага.

Чим далі сусід від БМУ, тим менше він дізнається. SOM повторюють другий крок для N ітерацій.

Нижче наведено діаграму вхідного вектора різних кольорів. Ці дані надходять до SOM, який потім перетворює дані на значення 2D RGB. Нарешті, він розділяє та класифікує різні кольори.



Малюнок 15 Схема роботи методу SOM

2.3.8. Мережі глибокої віри (DBN)

DBN — це генеративні моделі, які складаються з кількох рівнів стохастичних латентних змінних. Приховані змінні мають двійкові значення і часто називаються прихованими одиницями.

DBN — це стек машин Больцмана зі зв’язками між шарами, і кожен рівень RBM взаємодіє з попереднім і наступним рівнями. Мережі глибокої віри (DBN) використовуються для розпізнавання зображень, розпізнавання відео та даних захоплення руху.

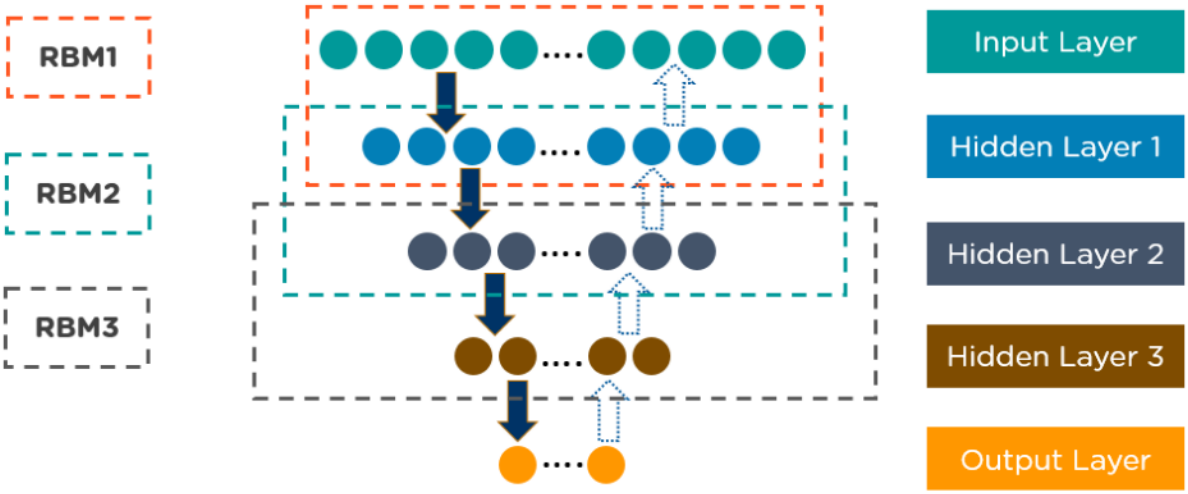
Як працюють DBN?

Алгоритми жадібного навчання навчають DBN. Алгоритм жадібного навчання використовує пошаровий підхід для вивчення генеративних ваг зверху вниз.

DBN виконують етапи вибірки Гіббса на двох верхніх прихованих шарах. На цьому етапі береться вибірка з RBM, визначена двома верхніми прихованими шарами.

DBN відбирають вибірку з видимих одиниць, використовуючи один прохід вихідної вибірки через решту моделі.

DBN дізнаються, що значення прихованих змінних на кожному рівні можна отримати за допомогою одного проходу знизу вгору.



Малюнок 16 Схема роботи методу DBN

2.3.9. Обмежені машини Больцмана (RBM)

Розроблені Джеффрі Гінтоном RBM — це стохастичні нейронні мережі, які можуть навчатися на основі розподілу ймовірностей за набором вхідних даних.

Цей алгоритм глибокого навчання використовується для зменшення розмірності, класифікації, регресії, спільної фільтрації, вивчення функцій і тематичного моделювання. RBM складають будівельні блоки DBN.

RBM складаються з двох шарів:

- Видимі одиниці
- Приховані підрозділи

Кожна видима одиниця пов'язана з усіма прихованими одиницями. RBM мають блок зсуву, який підключений до всіх видимих блоків і прихованих блоків, і вони не мають вихідних вузлів.

Як працюють RBM?

RBM мають дві фази: прохід вперед і прохід назад.

RBM приймають вхідні дані та перетворюють їх у набір чисел, який кодує вхідні дані в прямому проході.

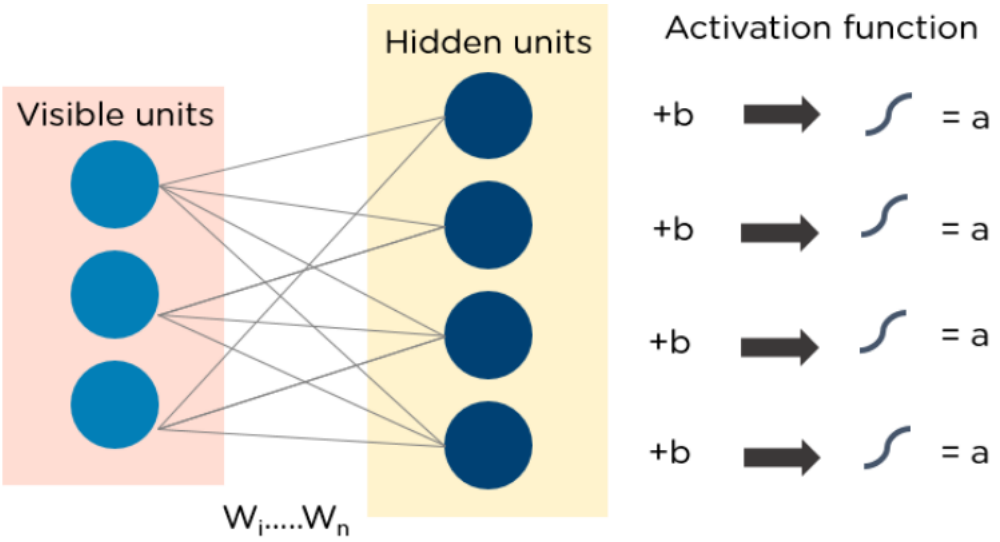
RBM поєднує кожен вхідний сигнал із індивідуальною вагою та одним загальним зміщенням. Алгоритм передає вихід на прихований шар.

У зворотному проході RBM беруть цей набір чисел і транслюють їх для формування реконструйованих вхідних даних.

RBM поєднує кожну активацію з індивідуальною вагою та загальним зміщенням і передає вихідні дані на видимий рівень для реконструкції.

На видимому шарі RBM порівнює реконструкцію з вихідним введенням, щоб проаналізувати якість результату.

Нижче наведено схему функціонування RBM:



Малюнок 17 Схема роботи методу RBM

2.3.10. Автокодери

Автокодери — це певний тип прямої нейронної мережі, у якій вхід і вихід ідентичні. У 1980-х роках Джеффрі Хінтон розробив автокодери для вирішення проблем навчання без нагляду. Це навчені нейронні мережі, які копіюють дані з вхідного рівня на вихідний. Автокодери використовуються для таких цілей, як відкриття фармацевтичних препаратів, прогнозування популярності та обробка зображень.

Як працюють автокодери?

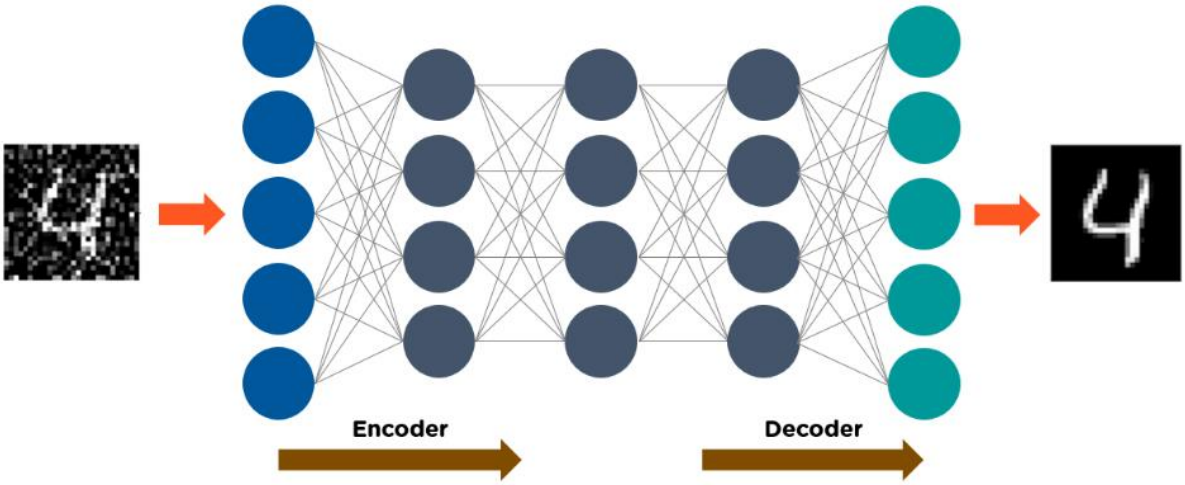
Автокодер складається з трьох основних компонентів: кодера, коду та декодера.

Автокодери побудовані так, щоб отримувати вхідні дані та перетворювати їх на інше представлення. Потім вони намагаються якомога точніше реконструювати вихідні дані.

Коли зображення цифри нечітко видно, воно надходить до нейронної мережі автокодера.

Автокодери спочатку кодують зображення, а потім зменшують розмір вхідних даних до меншого представлення.

Нарешті, автокодер декодує зображення для створення реконструйованого зображення.



Малюнок 18 Схема роботи автокодерів

Алгоритми глибокого навчання працюють майже з будь-якими типами даних і потребують великої кількості обчислювальної потужності та інформації для вирішення складних завдань.

2.4. Розгорнуто про CNN

2.4.1. Що таке штучний нейрон?

Штучний нейрон — це набір математичних операцій. Перш за все, вага та зміщення застосовуються уточненим способом до вхідного значення: у комп’ютерному зорі це значення пікселя. Потім до проміжного результату застосовується функція активації для представлення даних у просторі даних цієї функції. Часто це нелінійна функція активації, оскільки вона дозволяє представляти складні дані, де лінійна комбінація не працює.

2.4.2. Яку форму має архітектура згорткової нейронної мережі?

Аналіз поля зору здійснюваєть за допомогою сукупності субобластей, що перекриваються, які розміщують зображення. Кожен

субрегіон аналізується нейроном у мозку тварини для попередньої обробки невеликих обсягів інформації. Це називається згортковою обробкою.

Архітектура CNN створюється послідовно із блоків для виділення ознак, які розрізняють належний клас образу інших. Такі блоки складаються з одного або кількох:

- згорткових шари (CONV), які обробляють дані отримані із рецепторного поля;
- шарів корекції (ReLU);
- злиття шарів (POOL), який зменшує розмір проміжного зображення

Блоки змінюються один за одним, доки не дійдуть до кінцевого рівня мережі, який виконує класифікацію зображення та обчислення похибки між прогнозом і бажаним значенням:

- повністю зв'язаний (FC) шар, який є персептронним шаром;
- шар втрат (LOSS).

2.4.3. Як працює згортка

Зображення це:

- розрізати на підобласті, які називають плитками
- аналізується за допомогою згорткового ядра

Це згорткове ядро має розмір плитки, часто 3*3 або 5*5. Аналізована область (рецептивне поле) трохи більша за ядро, оскільки додається крок, щоб рецептивні поля перекривалися. Це дозволяє краще представити зображення та покращити когерентність його обробки.

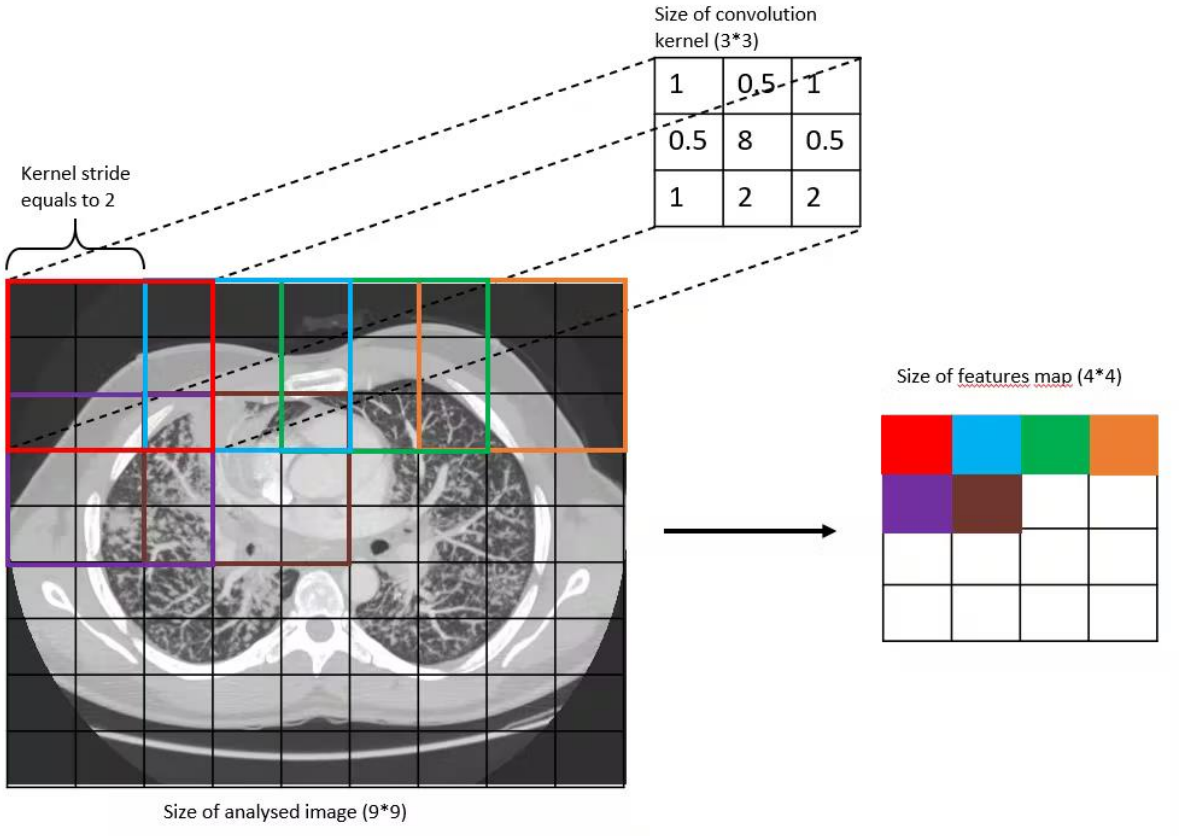
Аналіз особливостей зображення згортковим ядром є операцією фільтрації з вагами, які пов’язані з пікселями. Фільтрування зображення називається згорткою.

Після згортання виходить карта ознак, яка є абстрактним зображенням. Її значення залежать від змінних застосованого ядра згортки та значень пікселів вхідного зображення.

2.4.4. Що таке згортковий шар (CONV)

Згортковий шар — це стек згорток. Дійсно, якщо N згорткових ядер проходять через зображення, що зазвичай це призводить до N вихідних карт функцій. Кожне із згорткового ядра має властивості, специфічні для інформації, яка шукають на зображенні.

Вибір опцій ядра залежить від вибранної задачі. Під час глибокого навчання ці параметри автоматично вивчаються алгоритмом із навчальних даних. Функція втрат обчислює неточність між прогнозованим значенням і бажаним значенням.

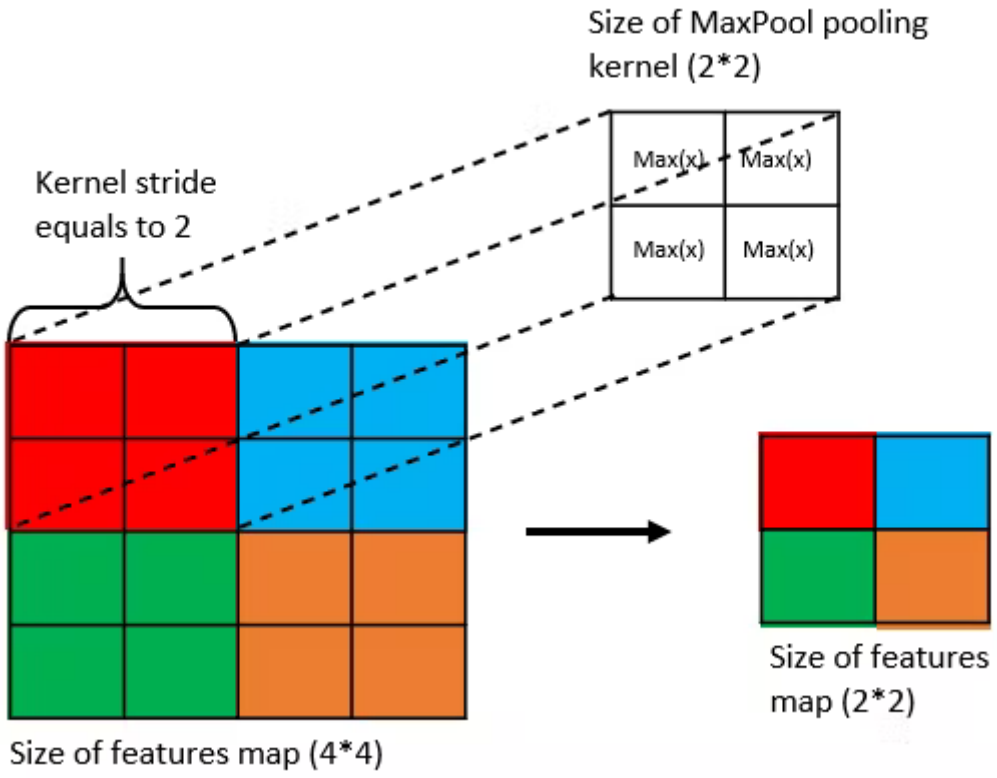


Малюнок 19 Діаграма згортки з розміром ядра 3*3 і кроком 2

2.4.5. Що таке рівень об’єднання (POOL)

Етап об’єднання є технікою підвибірки. Як правило, шар об’єднання вставляється через рівні проміжки між корекційним і згортковим шарами. Завдяки зменшенню розміру карт функцій і, отже, кількості мережевих параметрів, це прискорює час обчислення та зменшує ризик надмірної підгонки.

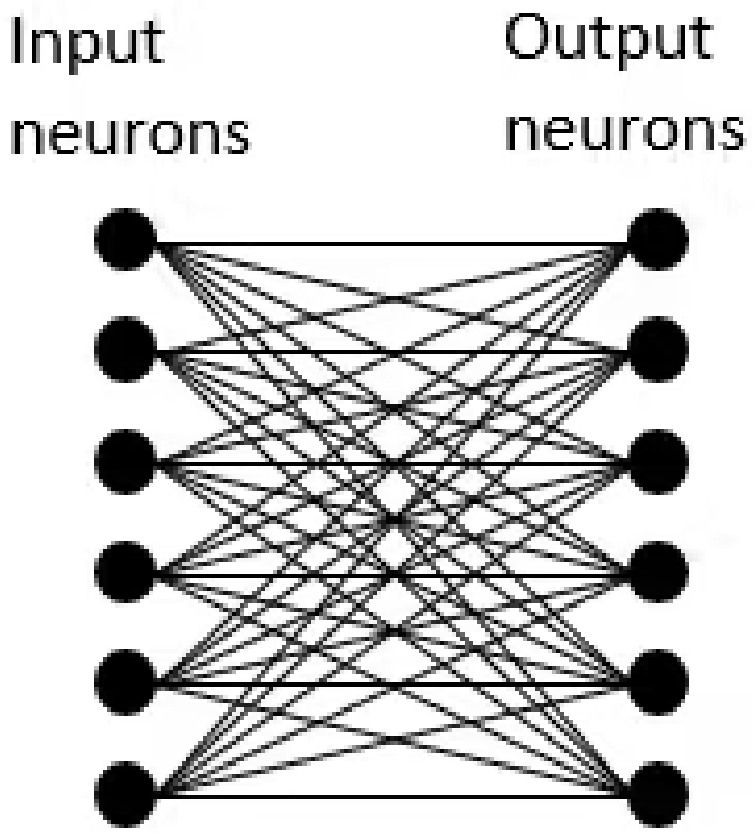
Найбільш поширеною операцією об’єднання є максимальна: MaxPool(2*2, 2). Він ефективніший, ніж середній, оскільки максимізує вагу сильних активацій. Він застосовується на виході попереднього шару як фільтр згортки розміру (2*2) і рухається з кроком 2. На виході шару об’єднання отримується характерна карта, стиснута в 4 рази.



Малюнок 20 Діаграма операції об’єднання з ядром MaxPool розміром 2*2 і кроком 2

2.4.6. Як працює «повністю підключений» (FC) рівень

Цей шар знаходиться в кінці мережі. Це дозволяє класифікувати зображення на основі характеристик, отриманих послідовністю блоків обробки. Він повністю зв'язаний, оскільки всі входи шару з'єднані з вихідними нейронами шару. Вони мають доступ до всієї вхідної інформації. Кожен нейрон присвоює зображенню значення ймовірності, що належить до класу і серед С можливих класів.



Малюнок 21 Діаграма повнозв'язного шару з 6 класами

Навпаки, фаза виділення ознак, де нейрони обробки незалежні один від одного, мають доступ лише до інформації рецептивного поля, яке вони обробляють.

2.4.7. Що таке рівень втрат (LOSS)

Рівень втрат є останнім рівнем мережі. Він обчислює похибку між прогнозом мережі та фактичним значенням. Під час завдання класифікації

випадкова змінна є дискретною, оскільки вона може приймати лише значення 0 або 1, причому 1 означає, що вона належить до класу, а 0 означає, що її немає. Ось чому найбільш поширеною та прийнятною функцією втрат є функція крос-ентропії.

Ця функція походить із галузі теорії інформації та вимірює глобальну різницю між двома розподілами ймовірностей (прогнозом моделі та реальним) для випадкової величини або набору подій. Формально написано:

$$loss(x,\ class) = -\sum_{class\ =\ 1}^{C} y_{x,\ class} \log\left(p_{x,\ class}\right)$$

Малюнок 22 Розрахунок рівня втрат

де y — розрахункова ймовірність того, що x належить до класу i, p — реальна ймовірність того, що x належить до класу i, враховуючи наявність класів C.

3. РОЗДІЛ 3 РЕЗУЛЬТАТИ РОБОТИ НЕЙРОННОЇ МЕРЕЖІ

У ході розробки нейроннох мережі було використанно алгоритм навчання CNN, за допомогою якого було створенно натренерованні моделі нейронних мереж на основі 1500 зображень в 10 категоріях за вмістом та 2 категоріях за дозвілом на публікацію. Розробка проводилася за допомогою існуючої бібліотеки для розробки нейронних мереж: Tensorflow.

3.1. Завантаження зображень для тренування та валідації

Для завантаження зображень завантаженних із інтернету, вікористовується функція image_dataset_from_directory(), яка отримує на вхід: директорію із зображеннями відфільтрованими за категоріями, зерно для змішування вихідних данних, розмір зображення форма якого буде отримана на виході, та число, яке регулює кількість зображень для тренування та валідації.

```
seed = random.randrange(10000)
train_ds = tf.keras.utils.image_dataset_from_directory(
    con.IMAGES_DIR,
    validation_split=0.3,
    subset="training",
    seed=seed,
    image_size=(con.IMG_SIZE, con.IMG_SIZE),
    batch_size=con.BATCH_SIZE)
val_ds = tf.keras.utils.image_dataset_from_directory(
    con.IMAGES_DIR,
    validation_split=0.3,
    subset="validation",
    seed=seed,
```

image_size=(con.IMG_SIZE, con.IMG_SIZE),

batch_size=con.BATCH_SIZE)

3.2. Створення моделі для обробки зображень

У першу чергу було додано рівень попередньої обробки, що масштабує вхідні значення до нового діапазону.

model = tf.keras.Sequential([

tf.keras.layers.Rescaling(1. / 255)

За алгоритмом роботи нейронної мережі на основі CNN, модель повинна вміщати у собі декілька згорткових рівнів. Ці рівені створюють ядро згортки, яке згортається разом із вхідними даними шару для створення тензора виходів. Якщо цей шар використовується як перший шар у моделі, потрібно вказати ключовий аргумент input_shape, у нашому випадку input_shape=(180, 180, 3) для зображень 180x180 RGB.

За кожним згортанним рівнем повинен бути рівень об’єднання. Цей рівень зменшує дискретизацію вхідного сигналу вздовж його просторових розмірів (висота та ширина), беручи максимальне значення у вхідному вікні (розміру, визначеному параметром pool_size) для кожного каналу вхідного сигналу. Вікно зсувається кроками вздовж кожного виміру.

tf.keras.layers.Conv2D(32, 3, activation='relu', input_shape=shape),

tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

tf.keras.layers.Conv2D(32, 3, activation='relu'),

tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

tf.keras.layers.Conv2D(32, 3, activation='relu'),

tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

Щоб завершити модель, потрібно передати останній вихідний тензор із згорткової бази в один або кілька щільних шарів для виконання

класифікації. Щільні шари приймають як вхідні дані вектори (які є одновимірними), а поточний висновок являє собою тривимірний тензор. Спочатку згладжується 3D-вихід у 1D, а потім додається один або кілька щільних шарів зверху. У данній роботі використовують данні із 10 та 2 вихідними классами, тому потрібно використати останній щільний шар із 10 а потім із 2 вихідними даними.

```
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(256, activation='relu',
                      kernel_regularizer=regularizers.l2(0.0001)),
tf.keras.layers.Dropout(0.5),


tf.keras.layers.Dense(num_classes)])
```

3.3. Налаштування, навчання та перевірка моделі

3.3.1. Оптимізатор

Перед початком тренування потрібно налаштувати модель.

```
model.compile(
  optimizer='adam',
  loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
```

Основним компоненто при налаштуванні є оптимізатор. Оптимізатори використовуються для покращення швидкості та продуктивності для навчання конкретної моделі. У данній моделі було використанно оптимізатор Adam.

Оптимізація Адама — це метод імовірнісного градієнтного зниження, який основується на адаптивному методі оцінці моментів перших двох порядків.

3.3.2. Функція втрат

Функцію втрат використовують, щоб визначити, наскільки прогнозовані значення відхиляються від фактичних значень у навчальних даних. З кожною епохою змініються ваги моделі, щоб втрати були мінімальними. У данній моделі використовується SparseCategoricalCrossentropy функцію втрат. Вона використовується, якщо є два або більше класів міток. Очікується, що мітки надаватимуться у представленні one_hot. Для кожної функції має бути N класів із плаваючою комою.

3.3.3. Метрики

При налаштуваннях моделі було використанно збір метрик моделі на основі *точності* та *втрат*

Метрика accuracy, обчислює, як часто передбачення збігаються з мітками.

Ця метрика створює дві локальні змінні, total і count, які використовуються для обчислення частоти, з якою y_pred відповідає y_true. Зрештою ця частота повертається як двійкова точність: ідемпотентна операція, яка просто ділить підсумок на кількість.

3.3.4. Навчання збереження моделі

Після налаштування моделі, можна починати навчання моделі. Метод model.fit() навчає модель для фіксованої кількості епох. Першим парамертом він приймає данні для тренування, які були отримані із методу tf.keras.utils.image_dataset_from_directory(). Також потрібно вказати кількість епох для начання у вигляці цілого числа, чим більше це чило тим більш кращим буде результат навчання.

Для визначення втрат та точності, які були вказані при налаштуванні, потрібно в параметр validation_data передати віповідні данні. Оцінка відбувається наприкінці кожної епохи. Модель не навчатиметься на цих даних.

У данній моделі були використанні зворотні виклики - спеціальні утиліти або функції, які виконуються під час навчання на певних етапах процедури навчання. Із заданними викликами, можно візуалізувати прогрес навчання, створити TensorBoard.

model.fit(train_ds, epochs=epoth, validation_data=val_ds,

callbacks=[tensorboard_callback, tensorboard, cm_callback])

Після начання, модель можно зберегти для подальшого використання у пов’язанних проектах.

model.save("saved_models/{}".format(NAME))

3.3.5. Модель ймовірності

Для перевірки моделі, потрібно створити модель ймовірності на основі навчиної моделі. Для цього потрібно прикріпити шар softmax для перетворення лінійних вихідних даних моделі - логітів - у ймовірності, які повинні бути легко інтерпретувати. Результати данної перевірки можно переглянути у підрозділі: «Зображення та аналіз ймовірностей»

probability_model = tf.keras.Sequential([model,

tf.keras.layers.Softmax()])

predictions = probability_model.predict(train_ds)

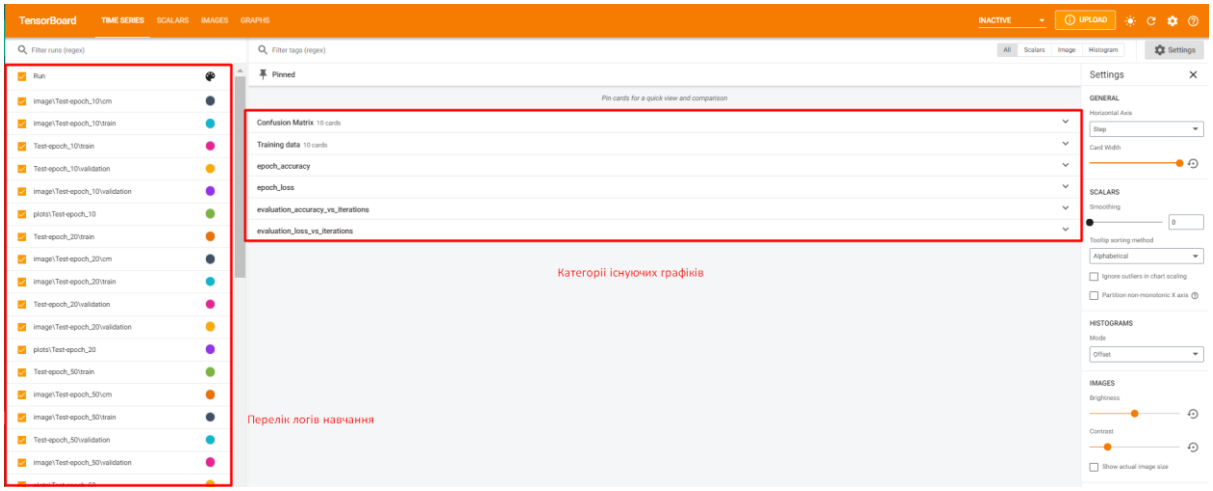
3.4. Графіки навчального процесу

Щоб проаналізувати ефективність навчання, потрібно мати можливість візуалізувати данні отримані при навчанні. Tensorflow надає таку можливість за допомогою вживеленого набір інструментів для візуалізації у вигляді TesorBoard.

За допомогою TensorBoard можна візуалізувати, необхідні дані для налаштування машинного навчання:

Інструменти, які надає TensorBoard:

- Моніторинг та відображення показників, таких як втрати та точність
- Проекція графа моделі
- Проеціювання вкладень у простір більш низької розмірності
- Демонстрація зображень, тексту та аудіо
- Збирання характеристик роботи програм TensorFlow



Малюнок 23 Графічна панель TensorBoard для перегляду зібраних метрик

У данній роботі відстужуються: точність, втрати, кореляцію та ймовірності, - як найважливіші для навчання та переналаштування моделі для більш прийнятливого результату. Для порівняння було узято дані із моделі з 10 категоріями за вмістом та 2 категоріями за дозвілом на публікацію, за період навчання 10 та 100 епох відповідно. Це було зробленно для порівняння якості навчання в залежності від періоду навчання та вихідних категорій.

3.4.1. Порівняння точності навчання

На двох нижче приведених графіках зображенно точність навчального та валідаційного процесу. Точність під час обробки зображень із 2 категоріями майже 100 відсотків, а із 10 категоріями 95

відсотків, що свідчить про залежніть якості навчання від кількості вихідних категорій.

Як видно з графіків точність валідації менша за точність під час навчання, це зумовленно перевірки точності на нових даних, які не враховуються під час навчання. Низька точність валізації свідчить про переоснащення моделі.

Щоб запобігти переоснащенню, найкращим рішенням є використання повніших навчальних даних. Набір даних повинен охоплювати весь діапазон вхідних даних, які, як очікується, оброблятиме модель. Тобто потрідно розширити кількість навчальних матеріалів. Модель, навчена на більш повних даних, природним чином узагальнює краще.

Коли це вже неможливо, наступним найкращим рішенням буде використання таких методів як регуляризація. Це накладає обмеження на кількість та тип інформації, яку може зберігати ваша модель. Якщо мережа може дозволити собі запам'ятати лише невелику кількість патернів, процес оптимізації змусить її зосередитись на найбільш помітних патернах, які мають більше шансів на гарне узагальнення.

У данній моделі було використано регулятор L2 та відсів для запобігання надлишкового переоснащення:

- kernel_regularizer=regularizers.l2(0.0001)

Додана вартість пропорційна квадрату значення вагових коефіцієнтів (тобто тому, що називається квадратом «норми L2» ваг). Регуляризация L2 також називається зменшенням ваги у контексті нейронних мереж.

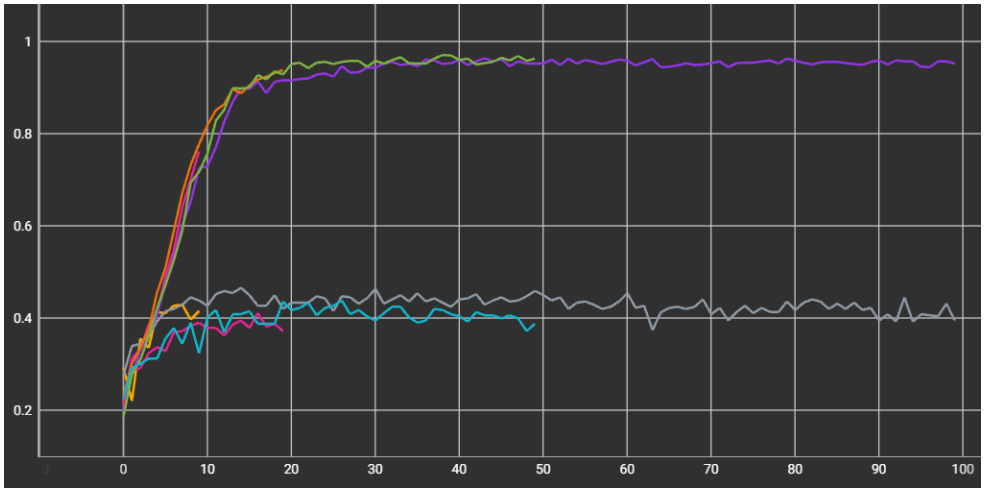
l2(0.001) означає, що кожен коефіцієнт у матриці ваги шару додасть 0.001 * weight_coefficient_value**2 до загальних втрат мережі.

- tf.keras.layers.Dropout(0.5)

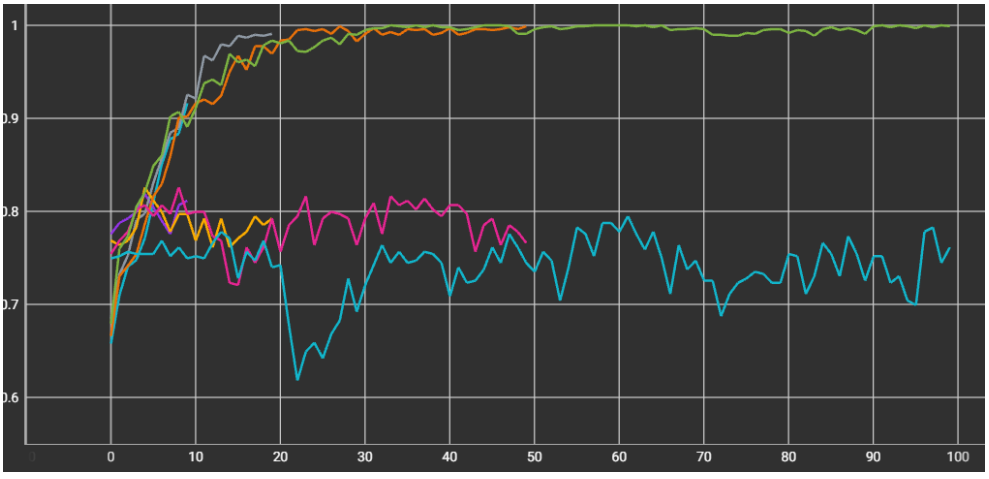
Пояснення відсіву полягає в тому, що оскільки окремі вузли в мережі не можуть покладатися на вихідні дані інших, кожен вузол повинен виводити функції, які корисні самі по собі.

"Показник відсіву" - це частка функцій, які обнулюються; зазвичай він встановлюється між 0,2 та 0,5. Під час тестування жодні одиниці не відкидаються, і натомість вихідні значення шару зменшуються на коефіцієнт, що дорівнює коефіцієнту відсіву, щоб збалансувати той факт, що активних одиниць більше, ніж під час навчання.

Також на переоснащення впливає кількість вихідних категорій. Як видно з другого графіка з 2 вихідними категоріями, валідаційна точність на 35 відсотків.



Малюнок 24 Точність навчання при 10 катигорій

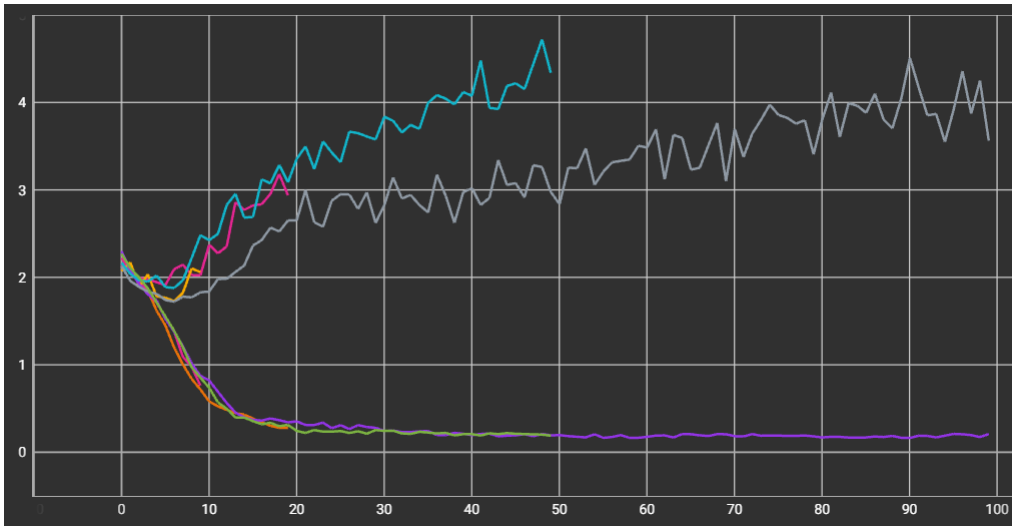


Малюнок 25 Точність навчання при 2 катигоріях

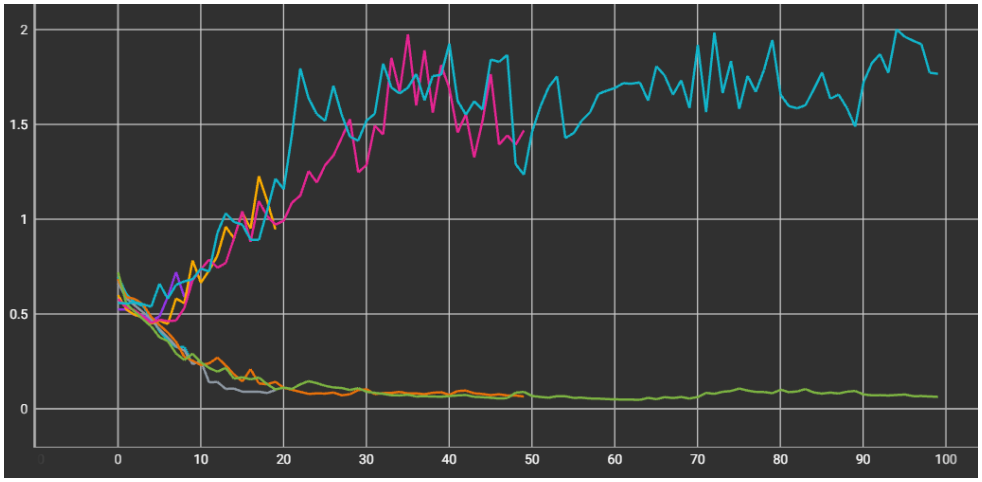
3.4.2. Втрати при навчанні

Функція втрат використовуєть, щоб визначити, наскільки прогнозовані значення відхиляються від фактичних значень у навчальних даних. Щоб втрати були мінімальними, потрібно змінювати вагові кофіцієнти. В цьому полягає суть навчання.

На графіках видно, що при 2 категоріях втрати під час валідації зменшилися майже у двоє, бо нейронніх мережі простіше фільтривати зображення по вихідним категоріям, коли їх мінімальна кількість.



Малюнок 26 Втрати при 10 категоріях



Малюнок 27 Втрати при 2 категоріях

3.4.3. Зображення та аналіз ймовірностей

Аналіз ймовірностей або ж прогнозування проводиться за допомогою завчасно підготовленного набору метеріалів. Прогнозування відноситься до вихідних даних алгоритму після того, як він був навчений на історичному наборі даних і застосований до нових даних при прогнозуванні ймовірності певного результату, у випадку даної дипломної роботи, чи відноситься зображення до однієї із категорій за вмістом зображення або чи можна публікувати дане зображення.

При вивченні діаграм можна помітити, що при невеликій кількості епох при збільшенні кількості категорій, зворстає ймовірність, що нейронна мережа помилиться при прогнованні категорії отриманного зображення. Так стається, бо зображення можуть містити зайві елементи, які будуть заважати проаналізувати до якої категорії віднести дане зображення. Також одним із чинників помилки можуть бути схожі за вмістом категорії, наприкладі категорій данної дипломної роботи: категорія bikini схожа на underwear, за напрямом використання чи boobs та chest, які відрізняються статью. Данна проблема дуже добре помітна при фільтруванні даних на основі категорій дозволу на публікацію тому, що категорії bikini та chest відносяться до категорії allowed, у той час як underwear та boobs до forbidden.

При більш детальному огляді діаграм можна помітити сірі стовпи. Вони показують процес прогнозування та можливості вибору на основі аналізу вмісту зображення. Це допомагає зрозуміти з якими саме категоріями при насчанні існують проблеми.

Малюнок 28 Перевірка за вмістом при 10 епохах

Малюнок 29 Перевірка на дозвіл при 10 епохах

Малюнок 30 Перевірка за змісттом при 100 епохах

Малюнок 31 Перевірка на дозвіл при 100 епохах.

ВИСНОВКИ З ДИПЛОМНОЇ РОБОТИ

Практичним значенням отриманих результатів є розробка моделі штучної нейронної мережі для розпізнавання зображень для ідентифікації сцен антисоціального характеру.

Було досліджено існуючі типи штучних нейронних мереж та особливості роботи при обробці зображень, проаналізованно методи навчання штучних нейронних мереж та їх практичне використання, знайденні слабкості при навчанні та валідаціх моделі підчас прогносування категорій на основі нового набору зображень із подальшим налаштуванням на основі отриманних результатів.

У подальшому, навчена модель буде використана при робці боту для модерації дискорд серверу, який буде фільтрувати завантажувані зображення із подальшим вибором, публікувати дане зображення чи ні. В разі заборони публікації, користувачу буде винесено покарання від рівня сцен антисоціального характеру.

ДЖЕРЕЛА

1. McMillan C. The Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network / C. McMillan, M.C. Mozer, P. Smolensky // Proc. XIII Annual Conf of the Cognitive Science Society, Hillsdale, NJ, USA. – 2001.

2. Кононюк А.Ю. Нейронні мережі і генетичні алгоритми [Текст] /]/ – Київ:Національний авіаційний університет, 2008. – 178с.

3. Рутковская, Д. Нейронные сети и генетические алгоритмы [Текст] / Д. Рутковская, М. Пипиньский, Л. Рутковский. – М.: Горячая линияТелеком, 2006. – 452 с.

4. TensorFlow: Відкрита платформа машинного навчання з відкритим кодом [Електронний ресурс] – Режим доступу: https://www.tensorflow.org/. – 04.06.2020

5. Keras [Електронний ресурс] – Режим доступу: https://keras.io/. – 05.06.2020.

6. NumPy. Фундаментальний пакет для обчислювань в Pyrhon. [Електронний ресурс] – Режим доступу: https://numpy.org/. – 02.04.2020

7. Aggarwal C. Neural Networks and Deep Learning. London: Springer International Publishing, 2018. 405 p.

8. Ніколенко С. І., Кадурін А. А., Архангельска Е. О. Глибоке навчання. Занурення у світ нейронних мереж. Пітер, 2016. 480 с

9. Хайкін С. Нейронні мережі: повний курс. Київ: ДіалектикаВільямс, 2018. 2-е вид. 1104 с.

10. Millstein F. Convolutional Neural Networks In Python: Beginner's Guide To Convolutional Neural Networks In Python. Berlin: CreateSpace Independent Publishing Platform, 2018. 121 p.

11. Zafar I., Tzanidou G. Hands-On Convolutional Neural Networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python. Boston: Packt Publishing, 2018. 272 p.

12. Gulli A., Sujit P. Deep Learning with Keras. Birmingham, UK: Packt Publishing. 2017. 318 p.

13. Agostinelli F., Hoffman F., Sadowski P. Learning Activation Functions to Improve Deep Neural Networks. Workshop paper contribution at the International Conference on Learning Representations (ICLR). 2015. P. 555-680.

14. Суцкевер, І .; Віньялс, О .; Ле, QV Послідовність навчання послідовності за допомогою нейронних мереж. У матеріалах щорічної конференції з нейронних систем обробки інформації 2014 р., Монреаль, QC, Канада, 8–13 грудня 2014 р .; С. 3104–3112.

15. Amari S. Field theory of self-organizing neural networks//IEEE Trans. Syst., Man, Cybern. 1983. V. 13. P. 741

ДОДАТОК А ЛІСТИНГ НЕЙРОННОЇ МЕРЕЖІ

```
import io
import itertools
import random

import numpy as np
import sklearn.metrics
import tensorflow as tf
from keras.callbacks import TensorBoard, LambdaCallback
from keras import regularizers
from matplotlib import pyplot as plt
import constans as con



def plot_image(i, predictions_array, true_label, img):
    try:
        true_label, img = true_label[i], img[i]
        plt.grid(False)
        plt.xticks([])
        plt.yticks([])

        plt.imshow(img.numpy().astype("uint8"))

        predicted_label = np.argmax(predictions_array)
        if predicted_label == true_label:
            color = 'blue'
        else:
            color = 'red'
```

```
        plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                             100 * np.max(predictions_array),
                                             class_names[true_label]),
                   color=color)
    except Exception as e:
        print(e)


def plot_value_array(i, predictions_array, true_label):
    true_label = true_label[i]
    plt.grid(False)
    plt.xticks(range(10))
    plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)
    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')


def plot_to_image(figure):
    buf = io.BytesIO()
    plt.savefig(buf, format='png')
    plt.close(figure)
    buf.seek(0)
    image = tf.image.decode_png(buf.getvalue(), channels=4)
    image = tf.expand_dims(image, 0)
    return image
```

```
def plot_confusion_matrix(cm, class_names):
    figure = plt.figure(figsize=(8, 8))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title("Confusion matrix")
    plt.colorbar()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names, rotation=45)
    plt.yticks(tick_marks, class_names)

    labels = np.around(cm.astype('float') / cm.sum(axis=1)[:, np.newaxis],
decimals=2)

    threshold = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        color = "white" if cm[i, j] > threshold else "black"
        plt.text(j, i, labels[i, j], horizontalalignment="center", color=color)

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    return figure


def log_confusion_matrix(epoch, logs):
    test_pred_raw = model.predict(train_ds)
    test_pred = np.argmax(test_pred_raw, axis=1)
    train_labels = np.concatenate([y for x, y in train_ds], axis=0)
```

```
    cm = sklearn.metrics.confusion_matrix(train_labels, test_pred)
    figure = plot_confusion_matrix(cm, class_names=class_names)
    cm_image = plot_to_image(figure)

    with file_writer_cm.as_default():
        tf.summary.image("Confusion Matrix", cm_image, step=epoch)


def image_grid():
    num_rows = 5
    num_cols = 4
    num_images = num_rows * num_cols

    figure = plt.figure(figsize=(2 * 2 * num_cols, 2 * num_rows))
    for images, labels in train_ds.take(1):
        for i in range(num_images):
            plt.subplot(num_rows, 2 * num_cols, 2 * i + 1)
            plot_image(i, predictions[i], labels, images)
            plt.subplot(num_rows, 2 * num_cols, 2 * i + 2)
            plot_value_array(i, predictions[i], labels)
        plt.tight_layout()

    return figure


for epoth in con.EPOCHS:

    seed = random.randrange(10000)
    train_ds = tf.keras.utils.image_dataset_from_directory(
```

```
    con.IMAGES_DIR,
    validation_split=0.3,
    subset="training",
    seed=seed,
    image_size=(con.IMG_SIZE, con.IMG_SIZE),
    batch_size=con.BATCH_SIZE)

val_ds = tf.keras.utils.image_dataset_from_directory(
    con.IMAGES_DIR,
    validation_split=0.3,
    subset="validation",
    seed=seed,
    image_size=(con.IMG_SIZE, con.IMG_SIZE),
    batch_size=con.BATCH_SIZE)

shape = ()
for x, y in train_ds.take(1):
    shape = x.shape

class_names = train_ds.class_names
num_classes = len(class_names)

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

model = tf.keras.Sequential([
    tf.keras.layers.Rescaling(1. / 255),
```

```
    tf.keras.layers.Conv2D(32, 3, activation='relu', input_shape=shape),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(32, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(32, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu',
                          kernel_regularizer=regularizers.l2(0.0001)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(num_classes)
])

model.compile(
    optimizer='adam',
    loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

NAME = "Test-epoch_{}".format(epoth)

tensorboard = TensorBoard(log_dir="logs/{}".format(NAME))

logdir = "logs/image/{}".format(NAME)
# Define the basic TensorBoard callback.
tensorboard_callback = TensorBoard(log_dir=logdir)
```

```
file_writer_cm = tf.summary.create_file_writer(logdir + '/cm')

cm_callback = LambdaCallback(on_epoch_end=log_confusion_matrix)

model.fit(train_ds, epochs=epoth, validation_data=val_ds,
          callbacks=[tensorboard_callback, tensorboard, cm_callback])

model.save("saved_models/{}".format(NAME))
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])

predictions = probability_model.predict(train_ds)

logdir = "logs/plots/{}".format(NAME)
file_writer = tf.summary.create_file_writer(logdir)

figure = image_grid()
with file_writer.as_default():
    tf.summary.image("Training data", plot_to_image(figure), step=0)
```