Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Розробка та дослідження методів класифікації образів одного класу в пошукових задачах
(тема)

Виконав:
студент 2 курсу, групи СШІм-18-1
Кійко К. В.
(прізвище, ініціали)
Спеціальність 122 – Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту (СШІ)
(повна назва освітньої програми)

Керівник проф. Філатов В. О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2019 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методів класифікації образів одного класу в пошукових задачах затверджена наказом по університету від 04 листопада 2019 р. № 1623Ст

2. Термін подання студентом роботи до екзаменаційної комісії ____ _____ 20__ р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки та дослідження методів класифікації образів одного класу в пошукових задачах, тестові виборки з UCT-репозіторію

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної області, задача класифікації, аналіз існуючих алгоритмів класифікації, методи оцінки алгоритмів класифікації, опис предметної області, потенціал платформи, можливості та ефективність методу, теоретичні дослідження, поняття машинного і глибинного навчання, глибинна нейронна мережа, принцип роботи нейронних мереж, нейронна мережа з точки зору математики, оновлення вагів нейронної мережі, згорткова нейронна мережа, порівняльна характеристика згорткових нейронних мереж, принцип роботи Faster R-CNN, поява Faster R-CNN, архітектура Faster R-CNN, базова нейронна мережа, VGG нейронна мережа, порівняння VGG і ResNet архітектур, якорі, принцип роботи RPN, об'єднання області інтересів, згорткова нейронна мережа на основі регіонів, оцінка, обгрунтування методів і засобів реалізації, мова програмування, середовище розробки PyCharm, веб-фреймворк Django, бібліотека Tensorflow, вимоги до розроблюваної системи, вимоги до системи в цілому, вимоги до структури та функціонування системи, вимоги до надійності, вимоги до функцій, виконуваних системою, підсистема збору, обробки, завантаження даних, підсистема зберігання

даних, підсистема роботи нейронної мережі, опис модулів програми, структура програми, принцип роботи програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1.1 – Приклад алгоритму дерева рішень, Рисунок 1.2 - Приклад нейронної мережі, Рисунок 1.3 - Приклад алгоритму k-найближчих сусідів, Рисунок 1.4 - Алгоритм перехресної перевірки, Рисунок 1.5 - Крива ROC, Рисунок 1.6 - Отримання інформації через зір, Рисунок 2.1 - Вузол нейронної мережі, Рисунок 2.2 - Схема примітивної нейронної мережі, Рисунок 2.3 - Модель нейронної мережі, Рисунок 2.4 - Параметри, які враховуються при розрахунку нейрона h1, Рисунок 2.5 - Розрахунок сигмоїда для o1, Рисунок 2.6 - Розрахунок сигмоїда для o2, Рисунок 2.7 - Частина нейронної мережі для розрахунку значення w5, Рисунок 2.8 - Частина нейронної мережі для розрахунку значення w6, Рисунок 2.9 - Частина нейронної мережі для розрахунку значення w1, Рисунок 2.10 - Залежність загальної помилки від h1,Рисунок 2.11 - VOC 2012 для Faster R-CNN, Рисунок 2.12 - COCO для Faster R-CNN, Рисунок 2.13 - VOC 2012 для R-FCN, Рисунок 2.14 - COCO для R-FCN, Рисунок 2.15 - Результати роботи SSD, Рисунок 2.16 - Результати роботи різних типів нейронних мереж,Рисунок 2.17 - Результати роботи SSD на наборі даних MS COCO, Рисунок 2.18 - VOC 2007 для YOLOv2, Рисунок 2.19 - Результат роботи YOLOv2 для набору даних PASCAL VOC 2012, Рисунок 2.20 -Результат роботи YOLO для MS COCO, Рисунок 2.21 - YOLOv3 для MS COCO, Рисунок 2.22 - Продуктивність для YOLOм3 з MS COCO, Рисунок 2.23 - Результат роботи FPN нейронної мережі для MS COCO, Рисунок 2.24 - Результат роботи RetinaNet нейронної мережі для MS COCO, Рисунок 2.25 - Швидкість навчання в порівнянні з точністю для RetinaNet на MS COCO,Рисунок 2.26 - Повна архітектура Faster R-CNN, Рисунок 2.27 - VGG архітектура, Рисунок 2.28 - Згортковий набір ознак (feature map), Рисунок 2.29 - Центри якорів по всьому вихідному зображенні, Рисунок 2.30 - RPN використовує згортковий набір ознак і генерує якоря на зображенні,Рисунок 2.31 - Згорткова реалізація архітектури RPN, де k - кількість якорів, Рисунок 2.32 - Об'єднання області інтересів, Рисунок 2.33 - Архітектура R-CNN, Рисунок 3.1 - Прогнози майбутнього трафіку для основних мов програмування, Рисунок 3.2 - Зростання популярності основних мов програмування, Рисунок 3.3 - Шаблон Model View Template (MVT), Рисунок 5.1 - Структура серверної частини, Рисунок 5.2 - Структура папки «migrations», Рисунок 5.3 - Структура папки «neuro_web», Рисунок 5.4 - Структура папки «object_similarity», Рисунок 5.5 - Структура папки «static», Рисунок 5.6 - Структура папки «templates»,

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	Дата
Основна частина	проф. Філатов В. О.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка

1	Отримання завдання на атестаційну роботу	04.11.19	виконано
2	Аналіз предметної області		виконано
3	Постановка завдання та узгодження з керівником		виконано
4	Дослідження методів кластерування компресії великих масивів		виконано
5	Розробка алгоритму		виконано
6	Розробка програмного забезпечення		виконано
7	Написання пояснювальної записки		виконано
8	Попередній захист		виконано
9	Захист перед ЕК		

Дата видачі завдання 04 листопада 2019 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Філатов В. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 87 с., 48 рисунка, 26 формул, 18 джерел.

Нейронні мережі та Інтернет технології кожного дня розвиваються. Вони стають невід’ємною частиною буденного життя та роботи: розваги, розумний дім, медицина, економіка та інші сфери лише невеликий приклад їх застосування. Тому, було прийнято рішення розробити систему для дослідження та розробки методів класифікації образів одного класу в пошукових задачах. Дана тема є актуальною, тому що розробками в даному напрямі вже зацікавились навіть такі великі компанії як Google, Samsung, IBM.

Як результат, буде представлений тестовий продукт, який продемонструє можливості продукту, а також зв’язок нейронних мереж та веб-технологій.

НЕЙРОННІ МЕРЕЖІ, ВЕБ-ТЕХНОЛОГІЇ, FASTER R-CNN, CNN, DEEP LEARNING, МАШИННЕ НАВЧАННЯ, PYTHON, DJANGO, JAVASCRIPT

РЕФЕРАТ

Пояснительная записка: 87 с., 48 рисунков, 26 формул, 18 источников.

Нейронные сети и Интернет-технологии ежедневно развиваются. Они становятся неотъемлемой частью повседневной жизни и работы: развлечения, умный дом, медицина, экономика и другие сферы являются лишь небольшим примером их применения. Поэтому было принято решение разработать систему для исследования и разработки методов классификации образов одного класса в поисковых задачах. Данная тема является актуальной, так как разработками в данном направление уже заинтересовались большие компании как Google, Samsung, IBM.

Как результат, будет представлен тестовый продукт, который продемонстрирует возможности продукта, а также связь между нейронными сетями и веб-технологиями.

НЕЙРОННЫЕ СЕТИ, ВЕБ-ТЕХНОЛОГИИ, FASTER R-CNN, CNN, ГЛУБИННОЕ ОБУЧЕНИЕ, МАШИННОЕ ОБУЧЕНИЕ, PYTHON, DJANGO, JAVASCRIPT

ABSTRACT

Explanatory note: 87 pages, 48 figures, 26 formula, 18 sources.

Neural networks and Internet technologies develop every day. They become an integral part of everyday life and work: rest, smart home, medicine, economics and other spheres are just a small example of their using. Therefore, it was decided to develop a system for research and development of methods for classifying images of one class in search problems. This theme is topical, since large companies such as Google, Samsung, IBM are already interested in developments in this area.

As a result, will be presented a test product that will demonstrate the capabilities of the product, as well as the connection between neural networks and web-technologies.

NEURAL NETWORK, WEB-TECHNOLOGIES, FASTER R-CNN, CNN, DEEP LEARNING, MACHINE LEARNING, PYTHON, DJANGO, JAVASCRIPT

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

CNN - згорткова нейронна мережа;

DL - глибинне навчання;

RPN - мережа пропозицій за регіонами;

SIMD - одиночний потік команд, множинний потік даних;

ANN - штучна нейронна мережа;

OCR - оптичне розпізнавання символів;

CV - комп'ютерний зір;

R-CNN - сверточная нейронна мережа, заснована на регіонах;

R-FCN - повністю сверточная мережа на основі регіонів;

SSD - однопоточний детектор;

FPN - мережа піраміди особливостей;

RetinaNet - єдина уніфікована мережа, яка складається з магістральної мережі та двох підмереж;

YOLO - ві дивитесь лише один раз, сучасна система виявлення об'єктів у реальному часі;

VGG - нейронна мережа Оксфордського університету для розпізнавання об'єктів на зображенні;

ResNet - скорочена назва часткової мережі, нова термінологія, яку вводить ця мережа, представляє собою залишкове навчання;

Inception - нейронна мережа компанії Google;

MobileNet - ефективні сверточные нейронні мережі для додатків мобільного зору;

IoU - (Intersection over Union) це оціночна метрика, яка використовується для виміру точності детектора об'єкта на конкретному наборі даних;

mAP - метрика для виміру точності детекторів об'єктів;

RoI Pooling - Об’єднання області інтересів, операція, яка широко використовується в задачах виявлення об’ктів з використанням сверточных нейронних мереж.

ВСТУП

Комп’ютери вміють вирішувати алгоритмічні та математичні задачі, але часто світ не може бути легко визначений за допомогою математичного рівняння. Розпізнавання облич та обробка мови - це лише деякі з проблем, які неможливо легко визначити в алгоритмі, але вони тривіальні для людини. Ключем до вирішення цієї проблеми є нейронні мережі, дизайни котрих дозволяє їм обробляти інформацію аналогічно нашому біологічному мозку, беручи за приклад роботу нашої нервової системи. Це робить їх корисними в задачах, наприклад, розпізнавання облич, як це робить наш мозок.

Нейронні мережі були створені ще в 70-х, 80-х роках минулого століття, але не могли бути в той час реалізовані через малі потужності обчислювальних машин. На сьогоднішній день - це одне з прогресивних вітвлень науки, яке постійно розвивається і все більше використовується в різних сферах життя і діяльності людини.

Ось лише деякі з них:

- медицина та охорона здоров’я: постановка діагнозу пацієнту, обробка зображень;

- геологічна розвідка: аналіз сейсмічних даних;

- економіка та бізнес: прогнозування курсу валют та цін;

- авіоніка: автопілоти;

- зв’язок: оптимізація сотових мереж;

- політологічні та соціологічні технології: передбачення результатів виборів;

Разом з наукою та технікою також швидко розвиваються інтернет-технології. Будь-яку необхідну інформацію можливо отримати зі всесвітньої павутини та зберегти її там.

Майже кожна велика компанія робить свій внесок в розвиток інтернет технологій та нейронних мереж.

Метою даної роботи є створення тестового варіанту веб-сервісу для розпізнавання об’єктів одного класу, а саме образів одягу та брендів.

Відповідно до поставленої мети необхідно розв’язати такі наукові задачі:

- аналіз існуючих методів та підходів до навчання штучних нейронних мереж;

- порівняння різних видів нейронних мереж та вибір оптимальної за результатами аналізу;

- створення веб-сервісу з використанням нейронних мереж.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Задача класифікації

Класифікація - це процес прогнозування класу для даних точок даних. Класи іноді називають цілями / мітками або категоріями. Класифікаційне прогностичне моделювання - завдання наближення функції відображення (f) від вхідних змінних (X) до дискретних вихідних змінних (y).

Наприклад, виявлення спаму у постачальників послуг електронної пошти можна визначити як проблему класифікації. Це бінарна класифікація, оскільки існує лише 2 класи: спам та не спам. Класифікатор використовує деякі навчальні дані, щоб зрозуміти, які задані вхідні змінні відносяться до класу. У цьому випадку в якості навчальних даних повинні використовуватися відомі спам- та неспам- листи. Коли класифікатор підготовлений точно, його можна використовувати для виявлення невідомої електронної пошти.

Класифікація належить до категорії наглядового навчання, де цілі також забезпечені вхідними даними. Існує багато застосувань для класифікації у багатьох сферах, таких як схвалення кредитів, медична діагностика, цільовий маркетинг тощо.

Є два типи учнів, які класифікуються як «ледачі» та «охочі».

«Ледачі» учні просто зберігають дані про навчання і чекають, поки з’являться дані тестування. Коли це відбувається, класифікація проводиться на основі найбільш суміжних даних, що зберігаються у навчальних даних. У порівнянні з «охочими» учнями, «ліниві» учні мають менше часу на навчання, але більше часу на прогнозування. Приклади алгоритмів: k-найближчих сусідів, міркування на основі конкретних випадків.

«Охочі» учні будують модель класифікації на основі даних про навчання, перш ніж отримувати дані для класифікації. Він повинен бути

спроможним взяти на себе єдину гіпотезу, яка охоплює весь простір примірника. Завдяки конструкції моделі, «охочими» учням знадобитися тривалий час на навчання і менше часу на прогнозування. Приклади алгоритмів: дерево рішень, наївний Байєс, штучні нейронні мережі.

1.2 Аналіз існуючих алгоритмів класифікації

Зараз існує маса алгоритмів класифікації, але неможливо зробити висновок, який з них перевершує інші. Це залежить від застосування та характеру наявного набору даних. Наприклад, якщо класи лінійно відокремлені, то лінійні класифікатори, такі як логістична регресія, лінійний дискримінант Фішера можуть перевершити складні моделі і навпаки.

Розглянемо деякі з них та оберемо той, який найкраще підійде для задачі класифікації образів одного класу в пошукових задачах.

Дерево рішень будує класифікаційні або регресійні моделі у вигляді структури дерева. Він використовує набір правил if-then, який є взаємовиключним та вичерпним для класифікації. Правила вивчаються послідовно, використовуючи дані тренувань по одному. Щоразу, коли правило засвоюється, кортежі, охоплені правилами, видаляються. Цей процес продовжується на навчальному наборі до досягнення умови припинення.

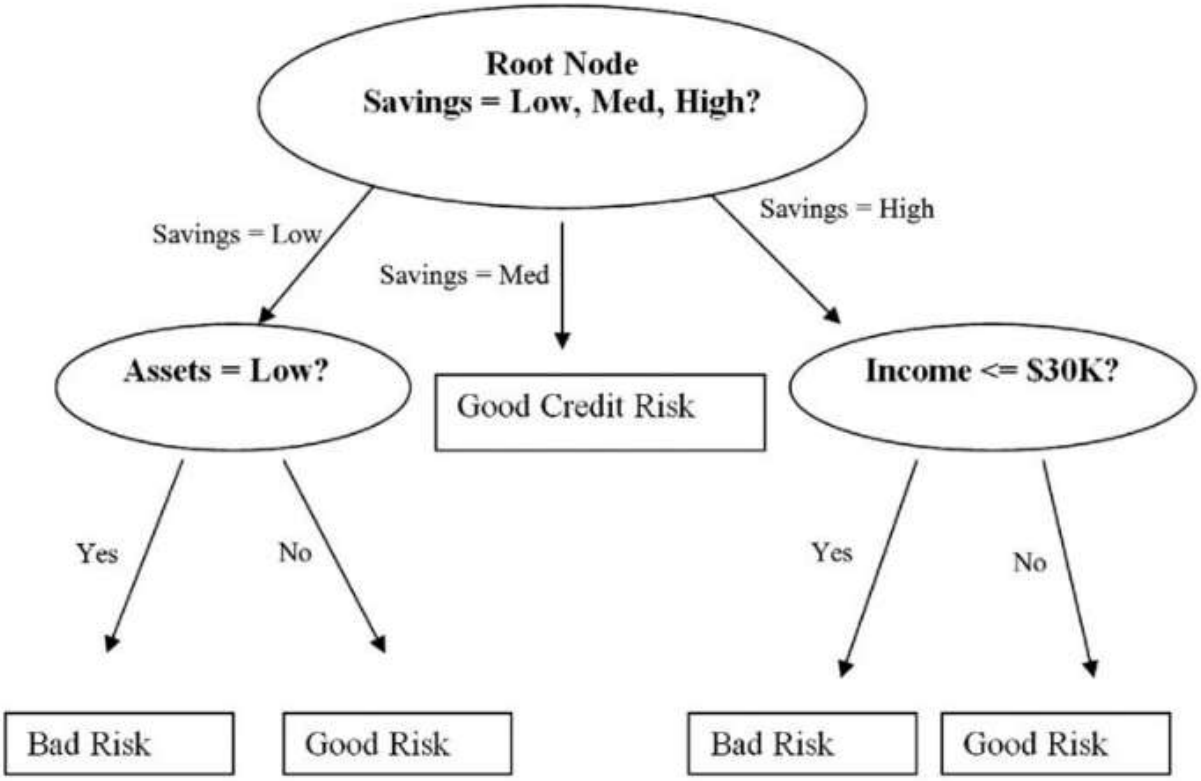

Рисунок 1.1 - Приклад алгоритму дерева рішень

Дерево побудовано рекурсивним способом розділення та перемоги зверху вниз. Усі атрибути повинні бути категоричними. В іншому випадку їх слід заздалегідь дискретизувати. Атрибути у верхній частині дерева мають більший вплив на класифікацію, і вони визначаються за допомогою концепції отримання інформації.

Дерево рішень може бути легко переобладнане, генеруючи занадто багато гілок, і може відображати аномалії внаслідок шуму або перешкод. Занадто встановлена модель має дуже низьку ефективність щодо небачених даних, хоча вона дає вражаючі показники щодо даних про навчання.

Переваги: здатний моделювати складні процеси прийняття рішень, інтуїтивно зрозумілі результати.

Недоліки: можна дуже легко переоцінити дані, будуючи дерево з гілками, які відображають інші частина в наборі даних. Способом боротьби з надлишковою обробкою є обрізка моделі, або не даючи їй

виростити зайві гілки (попередня обрізка), або видалити їх після побудови дерева (після обрізки).

Naive Bayes - імовірнісний класифікатор, натхненний теоремою Байєса за простим припущенням, атрибути яких умовно незалежні.

$$P(X|C_i) = \prod_{k=1}^{n} \quad P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i), \tag{1.1}$$

Класифікація проводиться шляхом виведення максимального заднього, який є максимальним $P(X|C_i)$, при цьому вище припущення застосовується до теореми Байєса. Це припущення значно знижує обчислювальні витрати, лише підраховуючи розподіл класів. Навіть незважаючи на те, що припущення є неправдивим у більшості випадків, оскільки атрибути залежні, надиво Байєс зумів виконати вражаюче.

Naive Bayes - це дуже простий алгоритм, який можна реалізувати, і в більшості випадків отримати хороші результати. Його можна легко масштабувати до більших наборів даних, оскільки це займає лінійний час, а не за рахунок дорогого ітеративного наближення, що використовується для багатьох інших типів класифікаторів.

Наївний Байєс підвержений проблемі, яка називається нульовою ймовірністю. Коли умовна ймовірність дорівнює нулю для конкретного атрибута, він не дає правильного прогнозу. Це потрібно чітко виправити за допомогою лапласівської оцінки.

Переваги: простий у реалізації та обчислювально легкий - алгоритм лінійний і не включає ітеративних обчислень. Хоча його припущення в більшості випадків не вірні, Naive Bayes напрочуд точний для великого набору проблем, масштабується до дуже великих наборів даних, і використовується для багатьох моделей NLP. Можна також використовувати для побудови багатошарових дерев рішень з класифікатором Байєса на кожному вузлі.

Недоліки: дуже чутливий до набору вибраних категорій, який повинен бути вичерпним. Проблеми, коли категорії можуть перетинатися або є невідомі категорії, можуть значно знизити точність.

Штучна нейронна мережа - це сукупність з'єднаних вхідних / вихідних одиниць, де кожне з'єднання має вагу, пов'язану з нею, розпочату психологами та нейробіологами для розробки та тестування обчислювальних аналогів нейронів. Під час фази навчання мережа вчиться, регулюючи ваги так, щоб можна було передбачити правильну мітку класу вхідних кортежів.

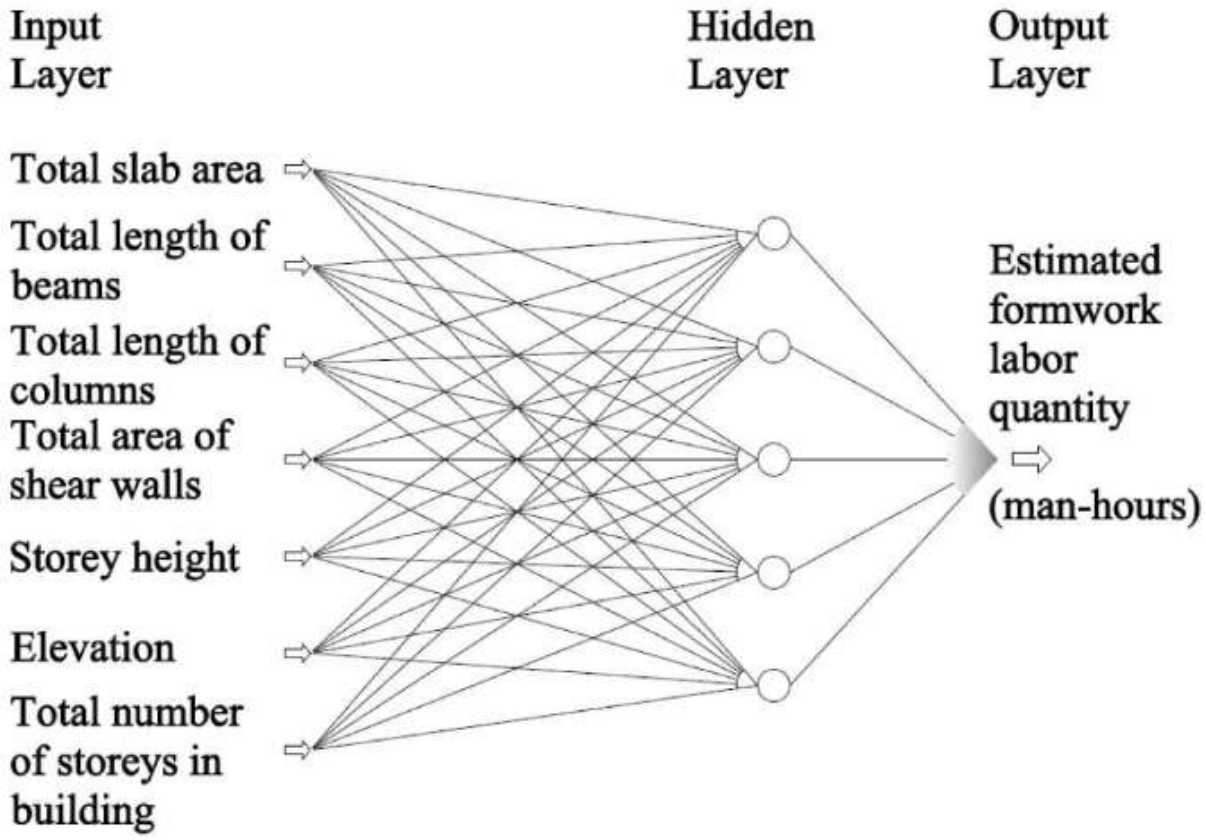

Рисунок 1.2 - Приклад нейронної мережі

На сьогоднішні день існує багато архітектур нейрнонних мереж, таких як Feed-forward, Convolutional, Recurrent тощо. Відповідна архітектура залежить від застосування моделі. У більшості випадків, Feed-forward моделі дають досить точні результати, особливо для задач обробки зображень згорткові мережі працюють краще.

У моделі може бути кілька прихованих шарів, залежно від складності функції, яка буде відображена моделлю. Наявність більшої кількості прихованих шарів дозволяє моделювати складні відносини, такі як глибинні нейронні мережі.

Однак, коли є багато прихованих шарів, потрібно багато часу для тренування і налаштування вагів. Іншим недоліком є погана інтерпретація моделі в порівнянні з іншими моделями, такими як Дерева рішень, через невідоме символічне значення вивчених ваг.

Але Штучні нейронні мережі працюють вражаюче в більшості реальних програм. Це висока точність з даними, які мають шуми і здатність класифікувати непідготовлені структури. Зазвичай Штучні нейронні мережі краще спрацьовують із постійними значеннями входів та виходів.

Переваги: дуже ефективні для задач з високою розмірністю, здатні обчислювати складні відносини між змінними, невичерпними наборами категорій і складними функціями, що стосуються введення змінних на вихід. Потужні параметри налаштування для запобігання надмірного та недостатнього тренування.

Недоліки: теоретично складні, важкі для впровадження. Не інтуїтивно зрозумілі і вимагають знань для налаштування. У деяких випадках для ефективності необхідна велика кількість даних для навчання.

k-найближчих сусідів - це «лінивий» алгоритм навчання, який зберігає всі випадки, що відповідають точкам навчальних даних у n-мірному просторі. Коли отримано невідомі дискретні дані, він аналізує найближчу k кількість збережених екземплярів (найближчих сусідів) і повертає найпоширеніший клас як прогнозування, а для даних з реальною цінністю повертає середнє значення k найближчих сусідів.

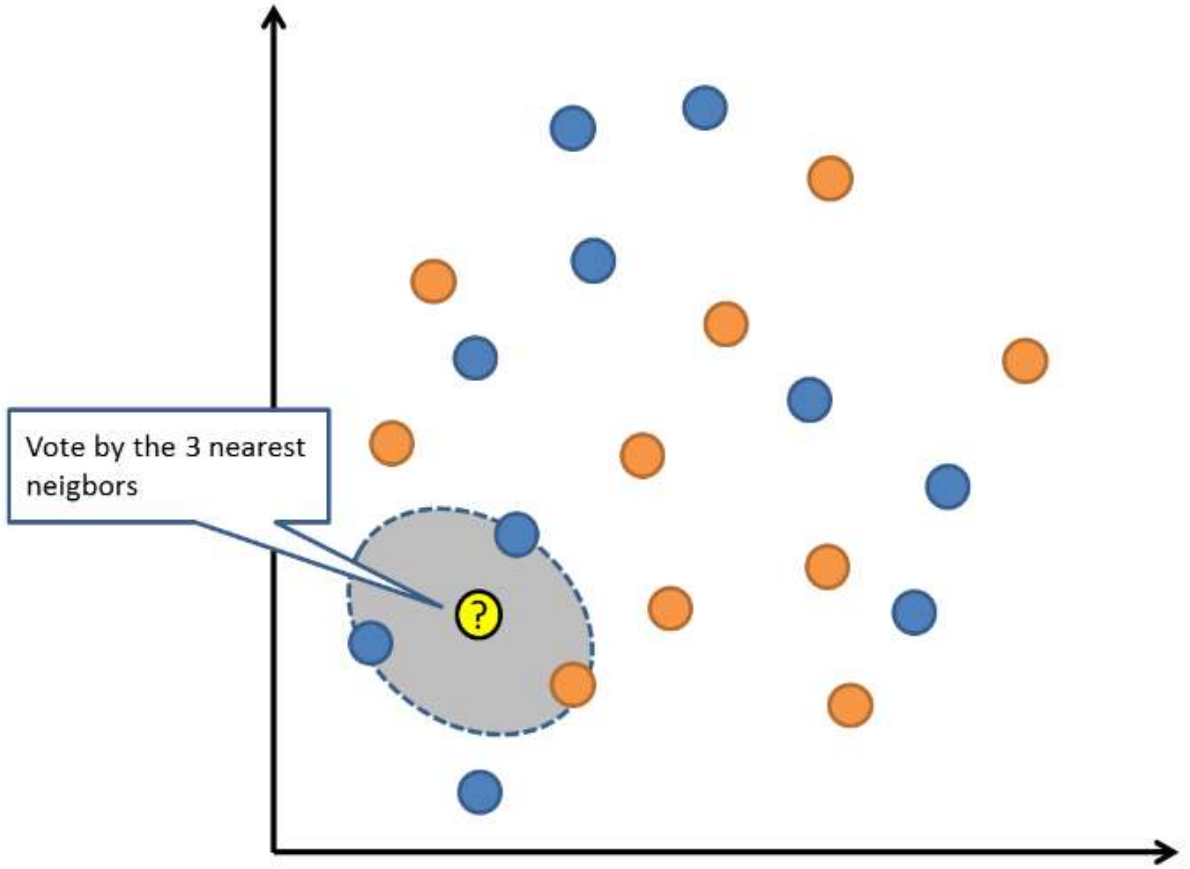

Рисунок 1.3 - Приклад алгоритму k-найближчих сусідів

В алгоритмі зваженого найближчого сусіда, на відстані він зважує внесок кожного з k сусідів відповідно до їх відстані, використовуючи наступний запит, що надає більшої ваги найближчим сусідам.

Формула для розрахунку відстані між сусідами:

$$w \equiv \frac{1}{d(x_q x_i)^2}, \tag{1.2}$$

Зазвичай алгоритм KNN надійний для даних в яких присутны шуми, оскільки він усереднює k-найближчих сусідів.

1.3 Методи оцінки алгоритмів класифікації

Після навчання моделі найважливішою частиною є оцінка класифікатора для перевірки його застосовності. Розглянемо деякі з поширених методів оцінки точності алгоритмів класифікації.

Існує кілька методів, і найпоширенішим методом є метод holdout. У цьому методі даний набір даних поділяється на 2 частини як тестовий та тренувальний, 20% та 80% відповідно. Тренувальний набір буде використовуватися для тренування моделі, а невідомі дані тестування будуть використані для перевірки його прогнозованої потужності.

Надмірне тренування - це поширена проблема в машинному навчанні, яка може виникати у більшості моделей. k-кратна перехресна перевірка може бути проведена для перевірки того, що модель не надто натренована. У цьому методі набір даних випадковим чином розподіляється на k взаємовиключних підмножин, кожна приблизно однакового розміру і одна зберігається для тестування, а інші використовуються для навчання. Цей процес повторюється протягом цілих k разів.

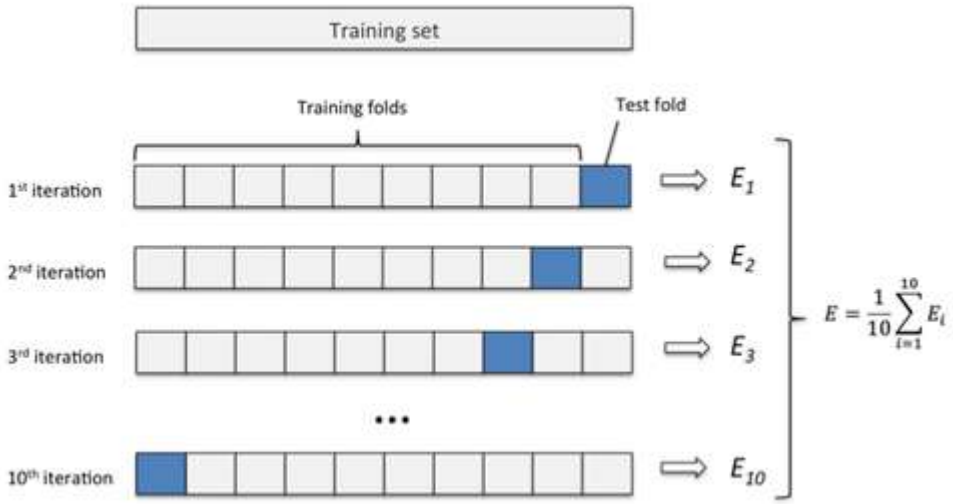

Рисунок 1.4 - Алгоритм перехресної перевірки

Точність - це частка відповідних екземплярів серед отриманих екземплярів, тоді як recall - це частка відповідних екземплярів, які були

отримані за загальну кількість відповідних екземплярів. Точність та recall використовуються як вимірювання відповідності.

Крива ROC (експлуатаційні характеристики приймача) використовується для візуального порівняння класифікаційних моделей, що показує компроміс між справжньою позитивною ставкою та помилковою позитивною ставкою. Площа під кривою ROC є мірою точності моделі. Коли модель ближче до діагоналі, вона менш точна і модель з ідеальною точністю матиме площу 1,0.

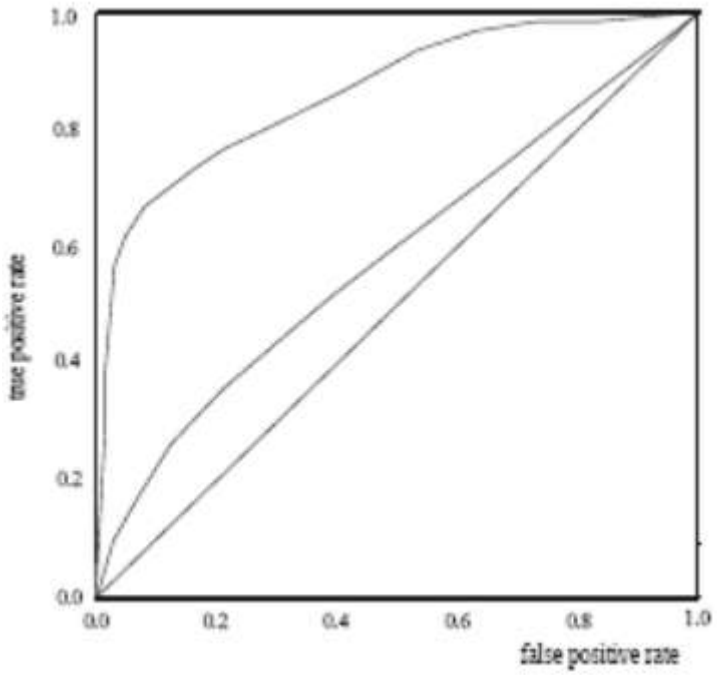

Рисунок 1.5 - Крива ROC

Провівши теоретичний аналіз різних алгоритмів класифікації, зробили висновок, що найкращим рішенням буде використовувати штучні нейронні мережі, які добре справляються з великим набором даних, мають високі потужності та найкращі результати для даного класу задач. Наступним кроком необхідно розглянути принципи роботи нейронних мереж та обрати нейронну мережу, а також її архітектуру, яка має найвищий показник точності в вирішенні задач класифікації образів в рамках одного класу в пошукових задачах.

1.4 Опис предметної області

Для вирішення поставленої задачі необхідно використовувати алгоритм комп’ютерного бачення (Computer Vision).

Computer Vision - це область Штучного інтелекту и комп’ютерних наук, ціль якої є надання комп’ютерам візуального розуміння світу за допомогою сукупності математичних алгоритмів Нayo.

Ціль Computer Vision - наслідувати зір людини з використанням цифрових зображень за допомогою трьох основних компонентів обробки, які виконуються один за одним:

1. Отримання зображення;
2. Обробка зображення;
3. Аналіз та розуміння зображення.

Оскільки наше людське візуальне сприйняття світу знаходить своє відображення в нашій властивості приймати рішення через те, що ми бачимо, надання такого візуального розуміння комп’ютерам дозволить їм мати таку ж силу:

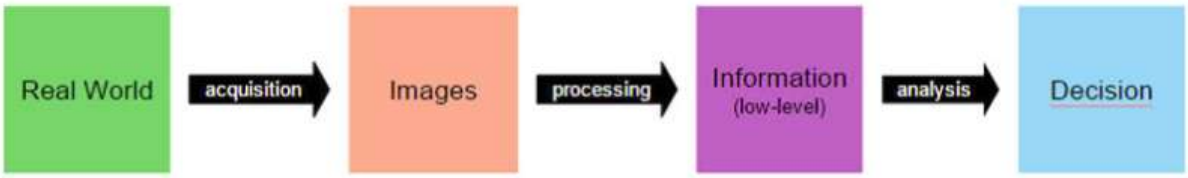

Рисунок 1.6 - Отримання інформації через зір

Мета системи - це отримання інформації, обробка та представлення результатів. Для досягнення поставленої мети необхідно використовувати технологію розпізнавання об’єктів.

Розпізнаванн об’єктів - це технологія комп’ютерного бачення для визначення об’єктів на фото та відео. Є результатом роботи алгоритмів машинного навчання.

Коли людина дивиться на зображення, вона помічає деякі об’єкти, людей, місця. В цьому полягає основна ідея алгоритму - навчити машину

«бачити» зображення як людина, щоб отримати розуміння того, що зображено на рисунку або фото.

Розпізнавання об’єктів - ключова технологія, яка лежить в основі автомобілів без водія, які дозволяють їм розпізнавати знаки на дорозі або відрізнити пішохода від стовпа. Він также корисний в різних застосуваннях, таких як ідентифікація захворювань на біо-зображенні, роботичному баченні.

Використовуючи технологію розпізнавання об’єктів можно впростити процесс пошуку образу заданого класу, що і лягло в основу даної роботи.

1.5 Потенціал платформи

На сьогоднішній день Інтернет все більше стає залежним від нейронних мереж. Майже на кожному великому ресурсі присутній штучний інтелект в тому чи іншому проявленні: чат-боти, розпізнавання тексту, облич і т. д.

Проаналізувавши різні Інтернет ресурсі, отримали висновок, що аналогів даному сервісу мало. Ось деякі з них:

- Clarifai;
- IBM Watsons.

Ці системи добре справляються зі своєю задачею, але вони охоплюють широкий спектр категорій. Представлений сервіс беду орієнтований лише на розпізнавання в рамках одного класу, що підвищить точність результатів. Сервіс може зайняти своє місце на ринку штучного інтелекту завдяки своїй швидкості та точності.

Платформа може бути цікавою з точки зору реклами або як повноцінний сервіс з розпізнавання наприклад одягу, що і було реалізовано в рамках дослідження.

1.6 Можливості та ефективність методу

Для розпізнавання образів в рамках одного класу необхідно буде використовувати нейронну мережу.

Штучні нейронні мережі є одним з основних інструментів, які використовують в машинному навчанні. Якщо розглядати саму назву, то «нейронна» частина назви вказує на те, що це система, яка надихалась структурою мозку, щоб відтворити те, що люди вивчають. Нейронні мережі складаються зі вхідних і вихідних рівней, а також, в більшості випадків, з прихованого рівня, який складається з блоків, котрі перетворюють введення в те, що може використовувати вхідний рівень. Вони відмінні інструменти для пошуку моделей, які занадто складні або численні для людини-програміста, щоб навчити машину розпізнавати.

Основними корисними на практиці можливостями нейронних мереж є наступні:

− Гнучкість структури: можна різними способами комбінувати елементи нейромережі (нейрони і зв'язки між ними). За рахунок цього на одній «елементній базі» і навіть всередині «тіла» одного нейрокомп'ютера можна створювати абсолютно різні обчислювальні схеми, підібрати оптимальний для конкретного завдання число нейронів і шарів мережі.

− Ефективне вибудовування нелінійного відображення (mapping) простору входів на простір вихідних сигналів.

− Можливість роботи при наявності великого числа неінформативних, надлишкових, шумових вхідних сигналів - попереднього їх відсіву робити не потрібно, нейросеть сама визначить їх мала придатність для вирішення завдання і може їх явно відкинути.

− Можливість роботи з корельованими незалежними змінними, з різнотипною інформацією (яка вимірюється в безперервно значних, дискретно значних, номінальних, булевих шкалах), що часто приносить затруднення методам статистики.

− Нейронна мережа одночасно може вирішувати кілька завдань на єдиному наборі вхідних сигналів - маючи кілька виходів, прогнозувати значення декількох показників. Часто це допомагає нейромережі побудувати більш адекватні або більш універсальні «внутрішні» - проміжні концепції (тому що потрібно, щоб всі ці проміжні розрахунки були придатні не для однієї, а для декількох завдань одночасно) і, внаслідок цього, підвищити точність рішення цих задач в порівнянні з рішеннями завдань по-окремо.

− Алгоритми навчання (або синтезу) накладають досить мало вимог на структуру нейромережі і властивості нейронів. Тому при наявності експертних знань або в разі спеціальних вимог можна цілеспрямовано вибирати вид і властивості нейронів, збирати структуру нейронної мережі вручну з окремих елементів і задавати для кожного з них потрібні характеристики або обмеження.

− Нейромережа може навчитися вирішення завдання, яку людина-експерт вирішує недостатньо точно (або для якої взагалі відсутній експерт). Навчена мережа далі може бути представлена у вигляді явного алгоритму розв'язання задачі, наприклад, у вигляді набору правил «якщо ..., то ...» - і вивчення цього алгоритму може дозволити людині отримати нові знання.

− Синтезована (навчена) нейронна мережа має стійкість до відмов окремих елементів (нейронів) і ліній передачі інформації в ній. За рахунок того, що навик рішення задачі «розмазаний» по мережі, не відбувається катастрофічного падіння точності рішення при виході з ладу кількох елементів системи. Можна застосовувати і спеціальні методи для підвищення відмовостійкості. Це буває потрібним при апаратних реалізаціях мереж - для забезпечення побудови надійних систем з ненадійних елементів.

− Висока потенційна паралельність обчислень (наприклад, одночасне паралельне функціонування нейронів деякого шару мережі) дозволяє ефективно задіяти можливості сучасної обчислювальної техніки (від

використання SIMD-команд до багатопоточності і багатопроцессорності) - що прискорює процеси нейро моделювання та / або дозволяє використовувати синтезовані моделі для вирішення задач реального часу.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

Розуміння останніх досягнень в області штучного інтелекту може здатися приголомшуючим, але насправді це зводиться до двох концепцій: машинне навчання і глибинне навчання. Ці терміни часто передаються способами, які можуть змусити їх здаватися взаємозамінними ключовими словами, тому важливо розуміти відмінності, щоб вибрати правильний алгоритм.

2.1 Поняття машинного і глибинного навчання

Визначення машинного навчання: «Алгоритми, які аналізують дані, вивчають ці дані, а потім застосовують те, що вони дізналися, щоб приймати обґрунтовані рішення».

Прикладом алгоритму машинного навчання є потокова передача музики на вимогу. Щоб служба приймала рішення про те, які нові пісні або артистів рекомендувати слухачеві, алгоритми машинного навчання пов'язують переваги слухача з іншими слухачами, у яких є схожий музичний смак.

Машинне навчання вирішує всілякі автоматизовані завдання і охоплює різні галузі, від фірм із захисту даних, які шукають шкідливе ПО до фінансування професіоналів, які шукають вигідні угоди. Вони призначені для роботи як віртуальні персональні помічники.

На практиці глибинне навчання - це всього лише підмножина машинного навчання. Він технічно є машинним навчанням і функціонує аналогічним чином (отже, терміни іноді вільно міняються), але їх можливості різні.

Базові моделі машинного навчання стають все краще і краще, незалежно від їх функції, але вони все ж деякі рекомендації. Якщо алгоритм машинного навчання повертає неточне передбачення, то інженер повинен втрутитися і внести корективи. Але за допомогою моделі

глибинного навчання алгоритми здатні самостійно визначати, чи є прогнози точними чи ні.

Модель глибинного навчання призначена для постійного аналізу даних з логічною структурою, аналогічною тому, як людина зробила б висновки. Для цього глибинне навчання використовує багатошарову структуру алгоритмів, званих штучною нейронною мережею (ANN). Конструкція ANN натхненна біологічної нейронною мережею людського мозку. Це робить машинний інтелект набагато більш здатним, ніж стандартна модель машинного навчання.

Чудовим прикладом глибинного навчання є AlphaGo від Google. Google створив комп'ютерну програму, яка навчилася грати в абстрактну настільну гру Go, гру, відому тим, що вимагає гострого інтелекту і інтуїції. Граючи проти професійних гравців Go, модель глибокого навчання AlphaGo навчилася грати на рівні, що не помічалося раніше в штучному інтелекті, і все, не сказавши, коли він повинен зробити певний крок (як це було б зі стандартною моделлю машинного навчання). Це викликало справжній переполох, коли AlphaGo переміг кількох всесвітньо відомих «майстрів» гри; машина не тільки могла зрозуміти складні і абстрактні аспекти гри, але і стала одним з найкращих гравців.

2.2 Глибинна нейронна мережа

Нейронні мережі являють собою набір алгоритмів, які вільно моделюються використовуючи структуру нейронів людського мозку, які призначені для розпізнавання закономірностей. Вони інтерпретують сенсорні дані за допомогою свого роду машинного сприйняття, маркування або кластеризації сировини. Зразки, які вони розпізнають, є чисельними, що містяться в векторах, в які повинні бути переведені всі реальні дані, будь то зображення, звук, текст або тимчасові ряди.

Нейронні мережі допомагають групувати і класифікувати. Можна розглядати їх як кластерний і класифікаційний рівень зверху даних, які ви

зберігаєте і якими керуєте. Вони допомагають згрупувати непомічені дані відповідно до подібності серед вхідних даних прикладу і вони класифікують дані, коли в них є маркований набір даних для навчання (нейронні мережі також можуть витягувати функції, які подаються на інші алгоритми кластеризації та класифікації, тому глибинні нейронні мережі - це як компоненти більших додатків машинного навчання з використанням алгоритмів навчання, класифікації і регресії підкріплення).

Результатами роботи нейронних мереж є мітки, які можуть застосовуватися до даних: наприклад, spam або not_spam в поштовому фільтрі, good_guy або bad_guy при виявленні шахрайства, angry_customer або happy_customer в управлінні взаємовідносинами з клієнтами.

Нейронні мережі можуть вирішувати безліч завдань. Ось деякі приклади:

– Задача класифікації - всі задачі класифікації залежать від маркованих наборів даних; тобто люди повинні перенести свої знання в набір даних, щоб нейрон міг вивчити кореляцію між мітками і даними. Це відомо як контрольоване навчання. Визначення обличчя людей, ідентифікування людей в образах, розпізнавання міміки (злість, радість), ідентифікування об'єктів в зображеннях (знаки зупинки, пішоходи, маркери доріжок і т. Д.), Розпізнавання жестів в відео, виявлення голосу, ідентифікування ораторів, транскрибування мови в тексті , розпізнавання почуття в голосах, класифікування тексту як спам (в листах) або шахраювання (в страхових випадках); розпізнавання настрою в тексті (відгуки клієнтів);

– Задача кластеризації (групування) - це виявлення подібності. Глибоке навчання не вимагає лейб для виявлення подібності. Навчання без лейб називається неконтрольованим навчанням. Немарковані дані - це велика частина даних в світі. Один закон машинного навчання полягає в наступному: чим більше даних алгоритм може навчати, тим точніше він буде. Таким чином, неконтрольоване навчання має потенціал для виробництва високоточних моделей. Пошук: порівняння документів,

зображень або звуків для отримання схожих предметів. Виявлення аномалій: зворотня сторона виявлення подібності виявляє аномалії або незвичайну поведінку. У багатьох випадках незвичайна поведінка сильно корелює з речами, які ви хочете виявити і запобігти, наприклад, з шахрайством;

– Завдання прогнозування. З класифікацією глибинне навчання здатне встановлювати кореляції між пікселями зображення і ім'ям людини - це статичне передбачення. Точно так, при наявності достатньої кількості правильних даних, глибинне навчання здатне встановити кореляції між поточними подіями і майбутніми подіями. Це може призвести до регресії між минулим і майбутнім. Майбутня подія схожа на лейбу в деякому сенсі. Глибинне навчання не обов'язково враховує час або те, що щось ще не відбулося. З огляду на часові ряди, глибинне навчання може читати рядок числа і прогнозувати число, яке, швидше за все, відбудеться. Апаратні збої (центри обробки даних, виробництво, транспорт), розбивки на здоров'я (інсульти, серцеві напади, засновані на життєво важливих статистиках і даних від носіння), відхилення клієнтів (прогнозування ймовірності того, що клієнт піде, на основі веб-активності і метаданих), оборот співробітників (те ж саме, але для співробітників). Чим краще прогнозувати, тим краще можливо запобігти і попередити. Як можна помітити, з нейронними мережами люди рухаються до світу менших несподіванок. Також, з'явилася можливість створювати більш розумних агентів, які об'єднують нейронні мережі з іншими алгоритмами, такими як посилення навчання для досягнення цілей.

2.3 Принцип роботи нейронних мереж

Глибинні нейронні мережі складаються з декількох вузлів. Шари зроблені з вузлів. Вузол - це місце, де відбувається обчислення, який спрацьовує, коли воно зустрічає достатні стимули. Вузол об'єднує вхідні дані з набором коефіцієнтів або вагів, які або посилюють, або зменшують

цей вхід, тим самим привласнюючи значення входів для задачі, яку алгоритм намагається вирішити. Ці ваги, які надходять на вхід, підсумовуються і сума передається через так звану функцію активації вузла, щоб визначити, чи буде і в якій мірі цей сигнал просуватися далі через мережі, щоб вплинути на кінцевий результат, наприклад, на акт класифікації.

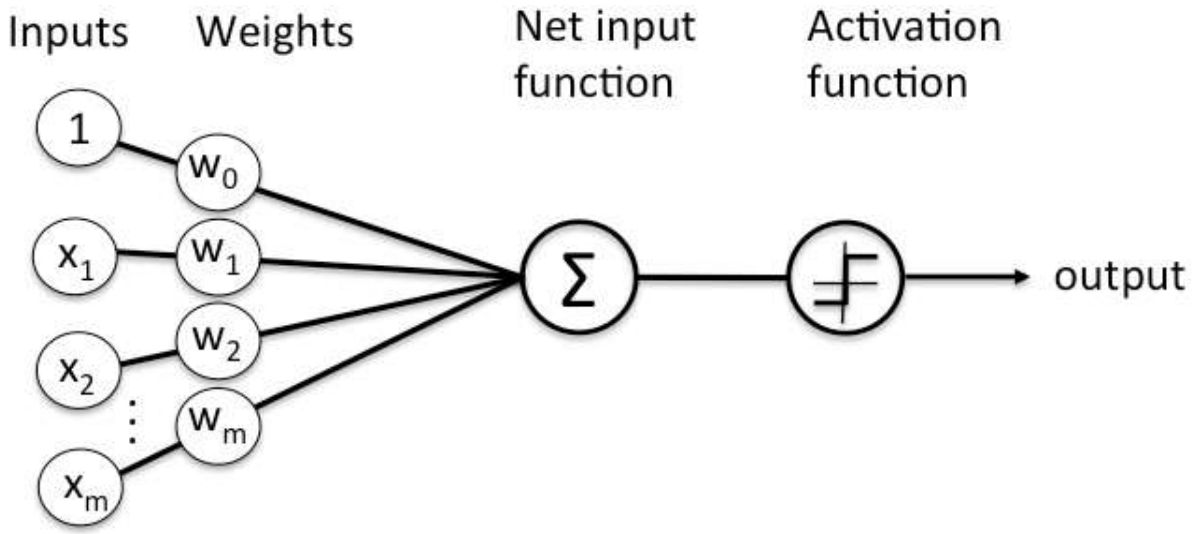

Рисунок 2.1 - Вузол нейронної мережі

Вузол - це ряд нейронних перемикачів, які вмикаються або вимикаються, коли вхідні дані подаються через мережу. Вихід кожного рівня одночасно є входом наступного рівня, починаючи з початкового рівня введення, який отримує дані.

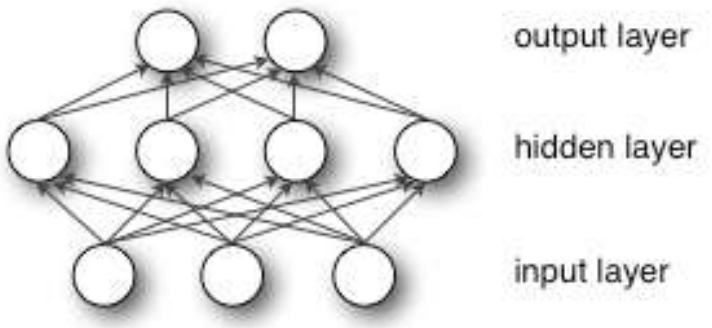

Рисунок 2.2 - Схема примітивної нейронної мережі

Нейронні мережі вивчають речі точно так, як правило, за допомогою процесу зворотного зв'язку, званого backpropagation (іноді скорочено «backprop»). Це включає в себе порівняння виведення, створюваного мережею, з виходом, який призначений для створення, і використання різниці між ними для зміни ваг з'єднань між одиницями в мережі, що працюють від вихідних блоків через приховані блоки до вхідних блоків. Іншими словами, з часом, backpropagation змушує мережу вчитися, зменшуючи різницю між фактичним і передбачуваним виходом до точки, де вони точно збігаються, тому мережу точно визначає те, що потрібно.

Для завдання розпізнавання об'єктів на зображенні вибрали Згорткову нейронну мережу (Convolutional Neural Network або CNN), так як вона найкраще підходить для вирішення завдань по розпізнаванню зображень.

2.4 Нейронна мережа з точки зору математики

Розглянемо математику нейронної мережі на прикладі простої мережі, у якій 2 входи (i_1, i_2), 1 прихований шар з 2 нейронами (h_1, h_2) і 2 виходи (o_1, o_2).

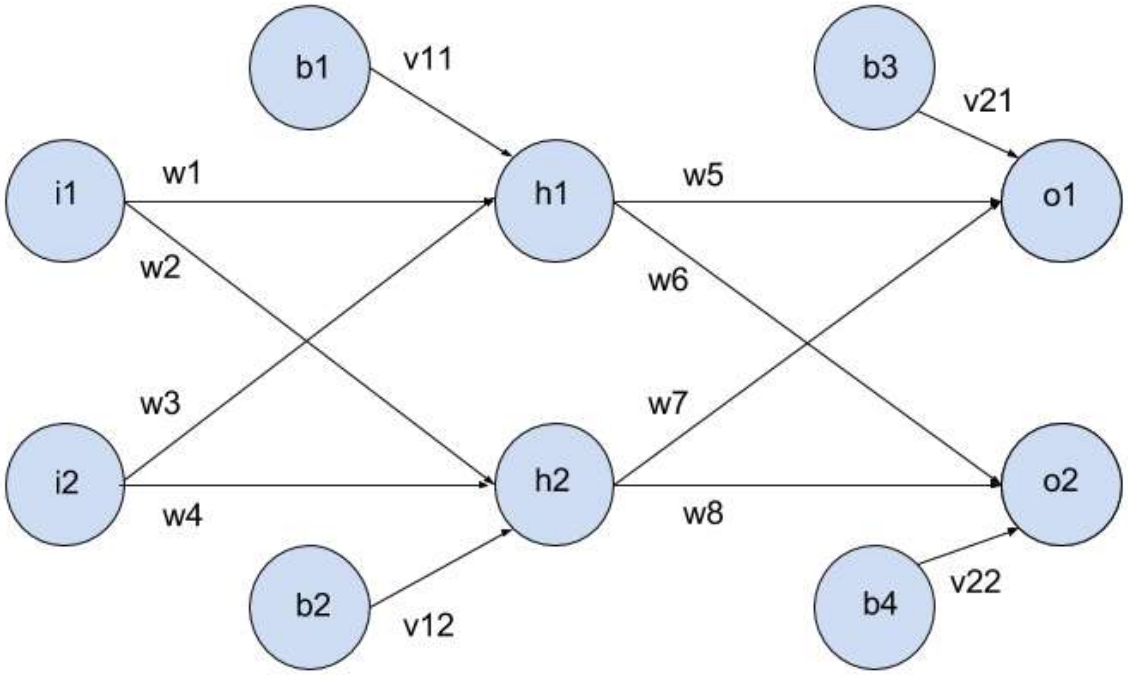

Рисунок 2.3 - Модель нейронної мережі

При створенні нейронної мережі, необхідно враховувати наступне (розглянемо на прикладі представленої мережі):

1) набір вхідних даних: 1, … , n (i_1, i_2);
2) набір результатів: 1, … , m (o_1, o_2);
3) число прихованих шарів (1);
4) число нейронів в кожному прихованому шарі (2 - h_1, h_2);

Число вхідних даних може відрізнятись від числа вихідних, кожен прихований шар може мати різне число нейронів.

Опціонально може зазначатися:

1) зміщення для кожного нейрона в прихованому шарі (шарах) (b_1, b_2) і результуючому шарі (b_3, b_4);
2) ваги зсувів в прихованому шарі (шарах) (v_{11}, v_{12}) і результуючому шарі (v_{21}, v_{22});
3) ваги з'єднуючих нейронів (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8).

Зсув необхідний для розрахунку нейрона. Після початку диференціювання, зсув не залежить ні від чого, тому він завжди дорівнює 0. Аналогічний чином відбувається для ваг зміщення.

У нейронної мережі, коли вона навчається, ваги оновлюються, тому намагаються якомога більше мінімізувати помилку між входами і виходами. Існує висока ймовірність отримання негативного значення. Також, результатом можете бути випадкове число від 0 до 1.

На першому етапі розглядаємо прихований шар. Розрахуємо значення h_1.

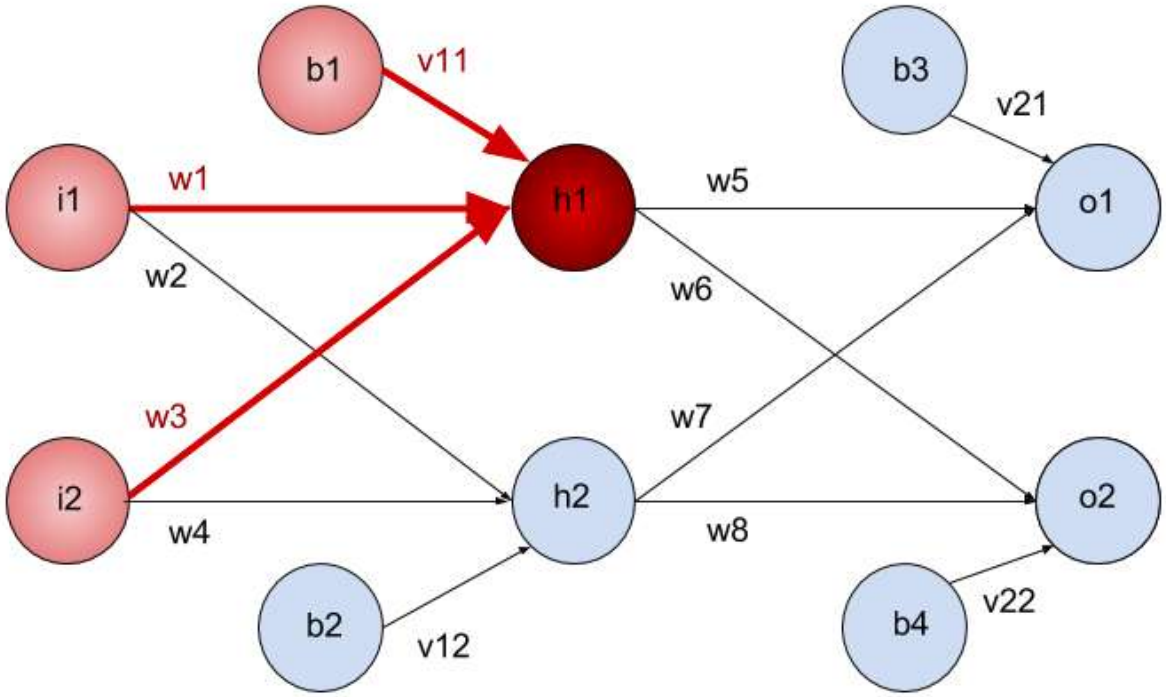

Рисунок 2.4 - Параметри, які враховуються при розрахунку нейрона h_1

Для розрахунку значення h_1 помножимо значення нейрона на його вагу. Формула для розрахунку одного нейрона:

$$(\sum_{m}^{n} \quad i_m w_m) + b_k v_k, \tag{2.1}$$

де i_m-вхідний нейрон;

b_k - зміщення;

w_m, v_k - ваги нейронів.

Для представленого прикладу, формула буде мати вигляд:

$$net\quad(h_1) = i_1 w_1 + i_2 w_3 + b_1 v_{11}, \tag{2.2}$$

Отримане значення використовуємо в функції сигмоїда. Результат сигмоїдного нейрона - плавний безперервний діапазон значень від 0 до 1. Вибір цієї функції обумовлений тим, що алгоритми навчання пов'язані з великою кількістю диференціювання, тому необхідно вибрати найпростішу дешеву для обробки функцію.

Сигмоїдна функція визначається як:

$$sig\quad(x) = \frac{1}{1 + e^{net\quad(x)}}, \tag{2.3}$$

Для нейрона h_1 , формула буде мати вигляд:

$$sig\quad(h_1) = \frac{1}{1 + e^{net\quad(h_1)}}, \tag{2.4}$$

Аналогічно проведемо розрахунки для h_2 , який залежить від i_1 і вагами w_2 , i_2 і вагами w_4 , b_2 вагами v_{12} .

Після того, як отримали значення сигмоїд для h_1і h_2 , ми можемо розрахувати результати для o_1і o_2 :

$$net\quad(o_1) = sig\quad(h_1) w_5 + sig\quad(h_2) w_7 + b_3 v_{21} \Rightarrow \tag{2.5}$$
$$sig\quad(h_1) = \frac{1}{1 + e^{net\quad(o_1)}},$$

$$net\quad(o_2) = sig\quad(h_1) w_6 + sig\quad(h_2) w_8 + b_4 v_{22} \Rightarrow \tag{2.6}$$
$$sig\quad(h_1) = \frac{1}{1 + e^{net\quad(o_2)}},$$

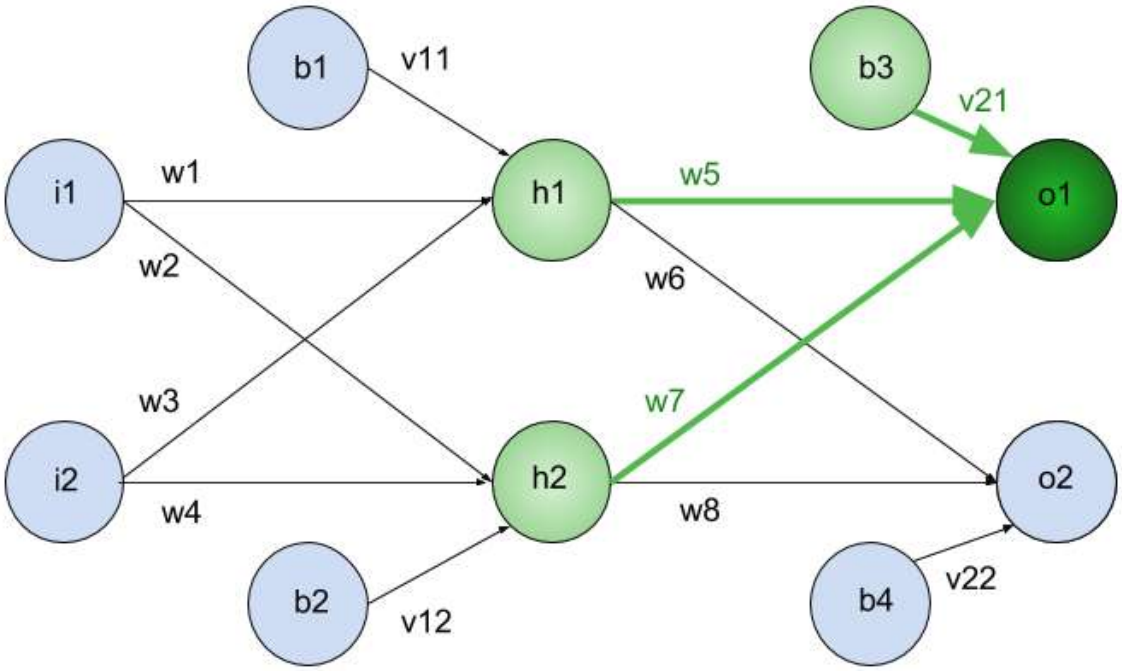

Рисунок 2.5 - Розрахунок сигмоїда для o_1

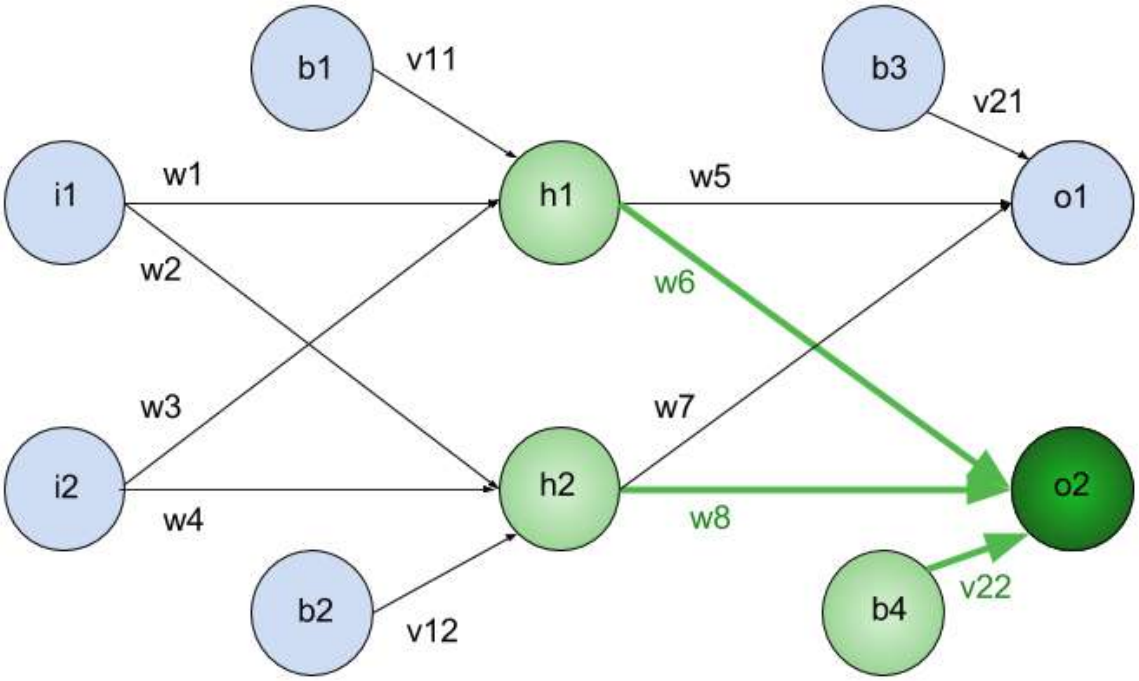

Рисунок 2.6 - Розрахунок сигмоїда для o_2

Порівняємо загальну помилку для сигмоїд ($sig\quad(o_1)$и $sig\quad(o_1)$) з результатами, які обрали ($target\quad(o_1)$і $target\quad(o_1)$), використовуючи Евклідову норму:

$$E_{total}\quad = \frac{1}{2}\sum_1^n\quad(target\quad(o_k) - sig\quad(o_k))^2, \tag{2.7}$$

$$E_{total}\quad = \frac{1}{2}(target\quad(o_1) - sig\quad(o_1))^2 + \frac{1}{2}(target\quad(o_2) - sig\quad(o_2))^2, \tag{2.8}$$

Проаналізуємо, як впливає $sig\quad(o_1)$на загальну помилку обчислень шляхом диференціювання. Права частина рівняння, після знака суми, не впливає на $sig\quad(o_1)$, тому ця частина буде дорівнює 0. Після цього, рівняння буде мати вигляд:

$$\frac{\partial E_{total}}{\partial sig\quad(o_1)} = sig\quad(o_1) - target\quad(o_1), \tag{2.9}$$

Аналогічно формула виглядає для $sig\quad(o_2)$:

$$\frac{\partial E_{total}}{\partial sig\quad(o_2)} = sig\quad(o_2) - target\quad(o_2), \tag{2.10}$$

Розрахували ці значення використовуючи часткове диференціювання, ігноруючи частину рівняння, яке не залежить від аргументу і правило ланцюга, щоб отримати наступне:

$$\frac{1}{2}(target\quad(o_2) - sig\quad(o_2))^2 \Rightarrow \tag{2.11}$$

$$sig\quad(o_2) - target\qquad(o_2),$$

Після цих кроків, відбуваються схожі розрахунки і перерахунки, з оновленням ваг, інших нейронів.

2.5 Оновлення вагів нейронної мережі

Метою цього розділу є побачити, як ваги впливають на загальну помилку.

Почнемо розглядати з w_5. Потрібно обчислити

$$\frac{\partial E_{total}}{\partial w_5}, \tag{2.12}$$

E_{total} залежить від $sig\quad(o_1)$. $sig\quad(o_1)$ залежить від $net\quad(o_1)$. $net\quad(o_1)$ залежить від w_5. Отже, E_{total} дійсно залежить від w_5.

Тому, щоб знайти, як E_{total} залежить від w_5, часткова похідна, яку нам потрібно обчислити:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial sig\quad(o_1)} \times \frac{\partial sig\quad(o_1)}{\partial net\quad(o_1)} \times \frac{\partial net\quad(o_1)}{\partial w_5}, \tag{2.13}$$

Щоб дізнатись, як E_{total} залежить від $sig\quad(o_1)$, обчислимо часткову похідну:

$$\frac{\partial E_{tota\quad l}}{\partial sig\quad(o_1)} = sig\quad(o_1) - target\qquad(o_1), \tag{2.14}$$

Щоб знайти, як $sig\quad(o_1)$ залежить від $net\quad(o_1)$, обчислимо часткову похідну сигмоїдної функції:

$$\frac{\partial sig\quad(o_1)}{\partial net\quad(o_1)} = sig\quad(o_1) \times (1 - sig\quad(o_1)), \tag{2.15}$$

Щоб знайти, наскільки $net\quad(o_1)$ залежить від w_5, обчислюємо часткову похідну:

$$\frac{\partial net\quad(o_1)}{\partial w_5} = sig\quad(h_1), \tag{2.16}$$

Після того, як отримали часткові похідні для всіх елементів рівняння, замінимо їх в рівнянні (2.13)

$$\frac{\partial E_{to\ tal}}{\partial w_5} = (sig\quad(o_1) - target\qquad(o_1)) \times (sig\quad(o_1) \times (1 - sig\quad(o_1))) \times sig\quad(h_1), \tag{2.17}$$

Тепер, коли підрахували, як w_5 впливає на загальну помилку, розглянемо нейронну мережу, яку ми моделюємо, зосередившись на w_5.

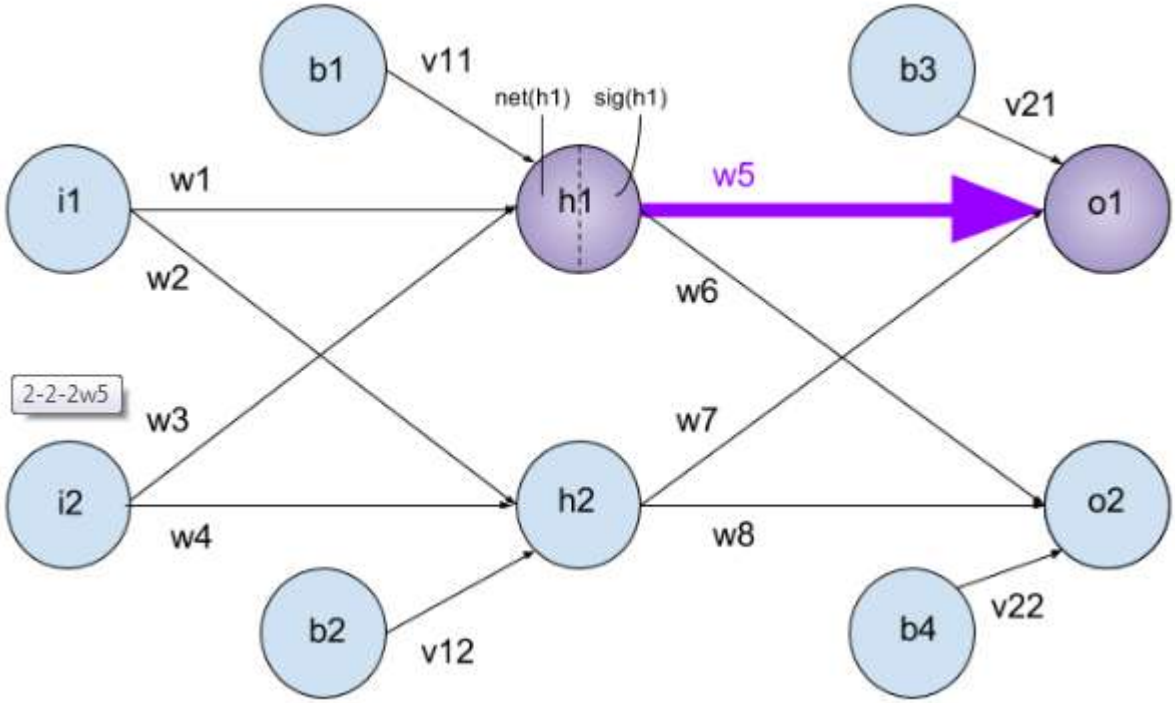

Рисунок 2.7 - Частина нейронної мережі для розрахунку значення w_5

Видно, що w_5 ефективно з'єднує сигмоїдне значення нейрона h_1 з нейроном o_1. Тому в обчисленні ми бачимо $sig\quad(o_1)$ і $sig\quad(h_1)$.

Нове значення w_5, w_{5_1}, зараз (де w_{5_0} - старе значення):

$$w_{5_1} = w_{5_0} - \eta \frac{\partial E_{total}}{\partial w_{5_0}}, \tag{2.18}$$

Де «η» - це ета, і саме там можна представляти рівень навчання. Чим вище рівень навчання, тим швидше нейромережа зменшить помилку, щоб наблизитися до результатів. Однак нейронна мережа буде менш точною. Зазвичай рівень навчання встановлюється на рівні 1/2.

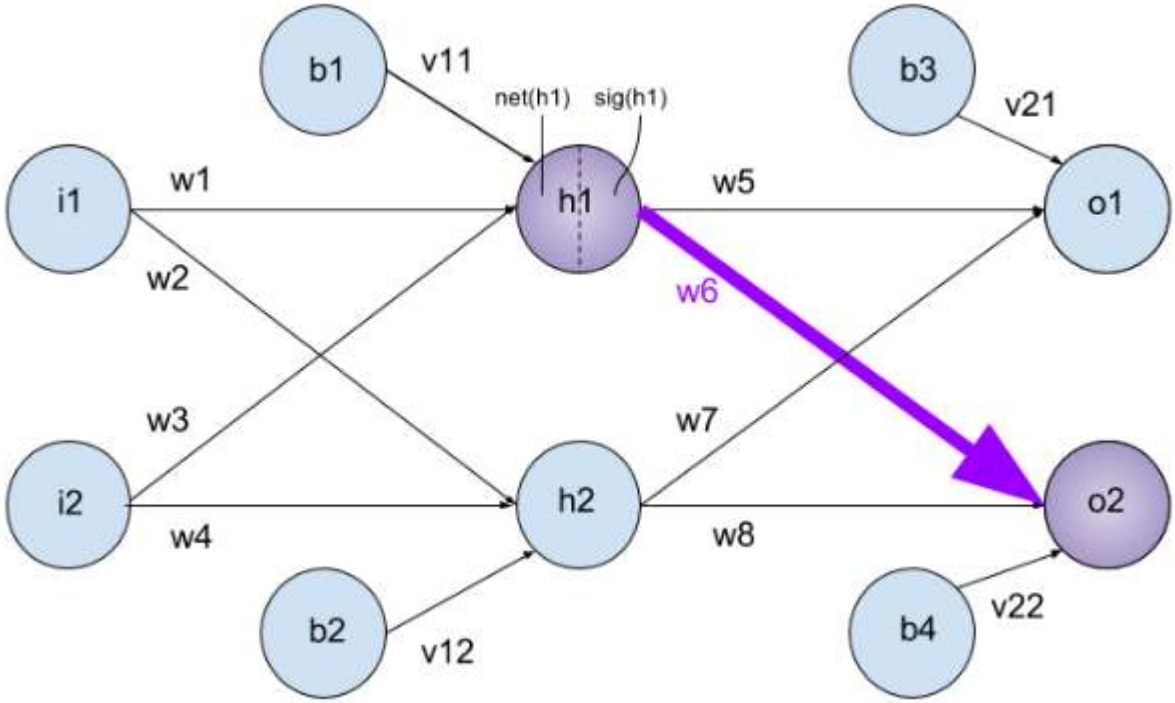

Рисунок 2.8 - Частина нейронної мережі для розрахунку значення w_6

Як бачимо на рисунку, w_6 з'єднує сигмоподібне значення нейрона h_1 з нейроном o_2. Тому рівняння для w_6, аналогічно як для w_5, матиме наступний вигляд:

$$\frac{\partial E_{total}}{\partial w_6} = (sig(o_2) - target(o_2)) \times (sig(o_2) \times (1 - sig(o_2))) \times sig(h_1), \quad (2.19)$$

Результат обчислюється у той самий спосіб, як з w_5 де: як E_{total} залежить від w_6, етотал залежить від $sig(o_2)$, який залежить від $net(o_2)$, який залежить від w_6.

На основі рівнянь, отриманих для w_5 та w_6, можемо отримати рівняння для w_7 та w_8:

$$\frac{\partial E_{total}}{\partial w_7} = (sig(o_2) - target(o_1)) \times (sig(o_1) \times (1 - sig(o_1))) \times sig(h_2), \quad (2.20)$$

$$\frac{\partial E_{total}}{\partial w_8} = (sig\quad(o_2) - target\qquad(o_2)) \times (sig\quad(o_2) \times (1 - sig\quad(o_2))) \times sig\quad(h_2), \tag{2.21}$$

Таким чином можемо розрахувати нові значення w_{6_1}, w_{7_1} та w_{8_1}:

$$w_{6_1} = w_{6_0} - \eta \frac{\partial E_{total}}{\partial w_{6_0}}, \tag{2.22}$$

$$w_{7_1} = w_{7_0} - \eta \frac{\partial E_{total}}{\partial w_{7_0}}, \tag{2.23}$$

$$w_{8_1} = w_{8_0} - \eta \frac{\partial E_{total}}{\partial w_{8_0}}, \tag{2.24}$$

Значення ета (η) є однаковим для кожної ваги у всій нейронній мережі (для ваг 1 - 8, а не лише 5 -8), але вони також можуть бути різними. Це означатиме, що ваги нейронної мережі навчаються з різною швидкістю, але для деяких моделей це може бути важливо. Якщо більше важливий один вихід, ніж інший (наприклад o_2 було важливішим). Рівень навчання w_6 та w_8 може бути вищим, ніж w_5 та w_7.

Розглянули, як w_5, w_6, w_7 і w_8 впливають на загальну помилку мережі і обчислили їхні нові значення. Але це лише один шар. Необхідно розглянути як w_1, w_2, w_3 і w_4 впливають на загальну помилку.

Спочатку розглянемо w_1.

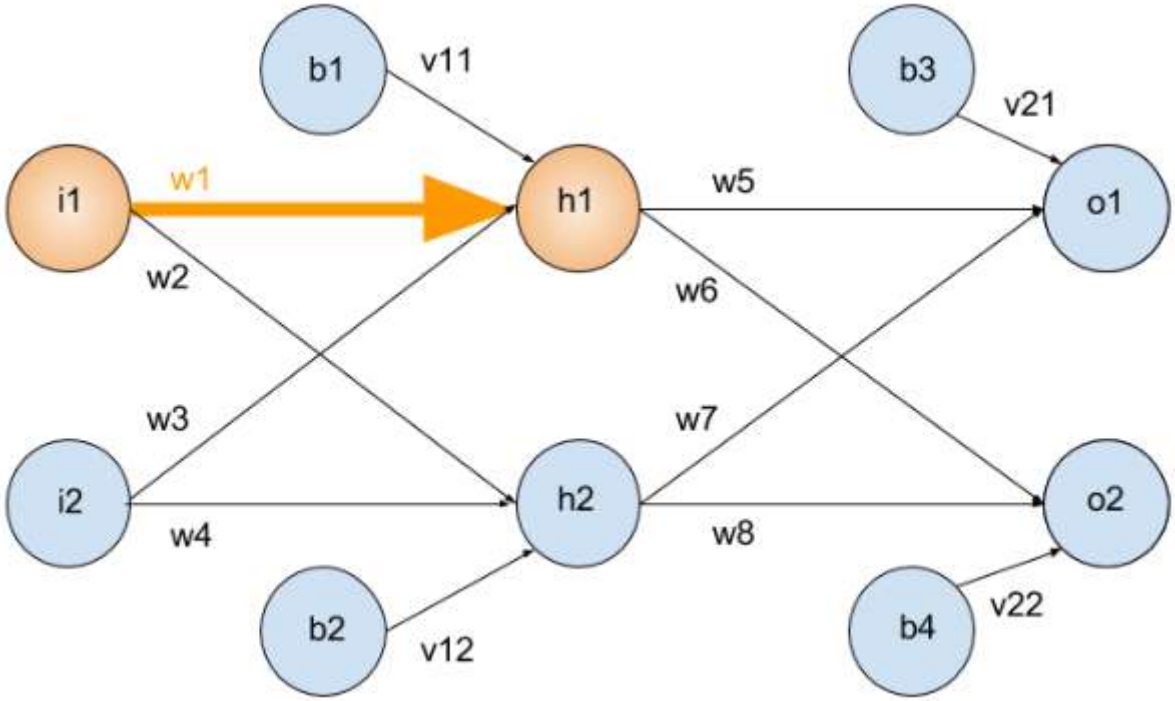

Рисунок 2.9 - Частина нейронної мережі для розрахунку значення w_1

Бачимо, що w_1 з'єднує i_1 і h_1. Спробуємо застосувати той же метод, який використовували для w_5і розглянемо як h_1 впливає на загальну помилку.

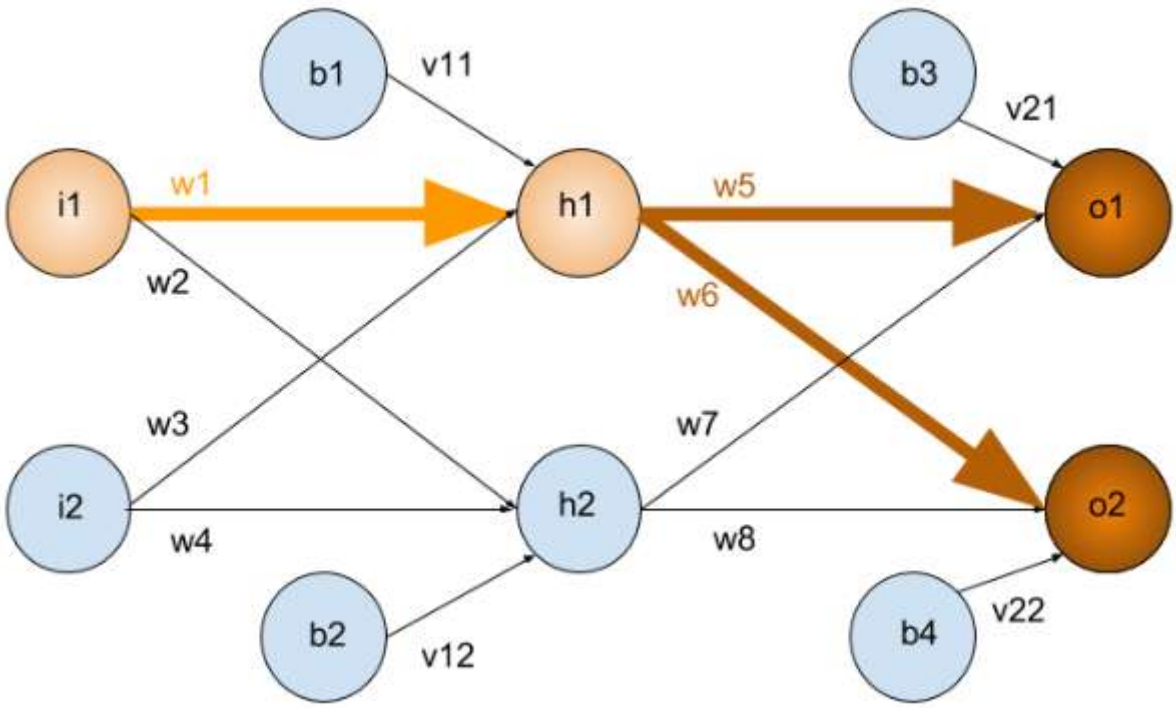

Рисунок 2.10 - Залежність загальної помилки від h_1

Знаємо, що $E_{tot\ \ al}$ залежить від $sig\ \ (o_1)$ та $sig\ \ (o_2)$. $sig\ \ (o_1)$ залежить від $net\ \ (o_1)$, а $sig\ \ (o_2)$ залежить від $net\ \ (o_2)$. $net\ \ (o_1)$ залежить від $sig\ \ (h_1)$, а $net\ \ (o_2)$ залежить від $sig\ \ (h_1)$. $sig\ \ (h_1)$ залежить від $net\ \ (h_1)$. В свою чергу, $net\ \ (h_1)$ залежить від w_1. На основі даної залежності можна вивести формулу:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial sig\ \ (h_1)} \times \frac{\partial sig\ \ (h_1)}{\partial net\ \ (h_1)} \times \frac{\partial net\ \ (h_1)}{\partial w_1}, \tag{2.25}$$

Де:

$$\frac{\partial E_{total}}{\partial sig\ \ (h_1)} = \tag{2.26}$$

$$(\frac{\partial E_{total}}{\partial sig\ \ (o_1)} \frac{\partial sig\ \ (o_1)}{\partial net\ \ (o_1)} \frac{\partial net\ \ (o_1)}{\partial sig\ \ (h_1)} + \frac{\partial E_{total}}{\partial sig\ \ (o_2)} \frac{\partial sig\ \ (o_2)}{\partial net\ \ (o_2)} \frac{\partial net\ \ (o_2)}{\partial sig\ \ (h_1)}),$$

Щоб дізнатись, як E_{total} залежить від $sig\ \ (h_1)$, обчислимо часткову похідну.

$$\begin{aligned} \frac{\partial E_{total}}{\partial sig\ \ (h_6)} = {} & ((sig\ \ (o_1) - target\ \ (o_1)) \times sig\ \ (o_1) \\ & \times (1 - (sig\ \ (o_1)) \times w_5) + \\ & ((sig\ \ (o_2) - target\ \ (o_2)) \times sig\ \ (o_2)(1 - \\ & (sig\ \ (o_2)) \times w_6)), \end{aligned} \tag{2.27}$$

Щоб дізнатися, як $sig\ \ (h_1)$ залежить від $net\ \ (h_1)$, обчислимо часткову похідну сигмоїдної функції.

$$\frac{\partial sig\ \ (h_1)}{\partial net\ \ (h_1)} = sig\ \ (h_1) \times (1 - sig\ \ (h_1)), \tag{2.28}$$

Щоб дізнатися, як $net\quad(h_1)$ залежить від w_1, обчислимо часткову похідну.

$$\frac{\partial net\quad(h_1)}{\partial w_1} = i_1, \tag{2.28}$$

Таким чином можна отримати формули обчислення для наступних відношень:

$$\frac{\partial E_{total}}{\partial w_2}, \frac{\partial E_{total}}{\partial w_3} and \quad \frac{\partial E_{total}}{\partial w_4}, \tag{2.29}$$

Використовуйте аналогічні формули зі швидкістю навчання, щоб знайти нові значення для w_1, w_2, w_3 і w_4.

$$w_{1_1} = w_{1_0} - \eta \frac{\partial E_{total}}{\partial w_{1_0}}, \tag{2.30}$$

$$w_{2_1} = w_{2_0} - \eta \frac{\partial E_{t\ otal}}{\partial w_{2_0}}, \tag{2.31}$$

$$w_{3_1} = w_{3_0} - \eta \frac{\partial E_{total}}{\partial w_{3_0}}, \tag{2.32}$$

$$w_{4_1} = w_{4_0} - \eta \frac{\partial E_{total}}{\partial w_{4_0}}, \tag{2.24}$$

На даному етапі розрахували всі ваги для представленої нейронної мережі. Слідуючи цим правил можно розрахувати ваги для нейронної мережі з будь-якою кількістю шарів.

2.6 Згорткова нейронна мережа

Згорткові нейронні мережі являють собою глибокі штучні нейронні мережі, які використовуються, перш за все, для класифікації зображень (наприклад, називати те, що вони бачать), згрупувати їх за подобою (пошук фотографій) і виконувати розпізнавання об'єктів усередині сцен. Це алгоритми, які можуть ідентифікувати особи, індивіди, вуличні знаки, пухлини, platypus і багато інших аспектів візуальних даних.

Згорткові мережі виконують оптичне розпізнавання символів (OCR) для оцифровки тексту і роблять обробку природною мовою можливої для аналогових і рукописних документів, де зображення є символами, що підлягають розшифровці. CNN також можуть застосовуватися до звуку, коли візуально представляються у вигляді спектрограми. Зовсім недавно згорткові мережі були застосовані безпосередньо в текстовій аналітиці, а також до графічних даними зі згортковими мережами графів.

Ефективність згорткових мереж (ConvNets або CNN) при розпізнаванні зображень є однією з основних причин, по якій вони користуються великою популярністю. Вони призводять до великих досягнень в області комп'ютерного зору (CV), який має очевидні додатки для самостійних автомобілів, робототехніки, безпілотних літаків, безпеки, медичних діагнозів і лікування для людей з ослабленим зором.

Але разом з цим, CNN дуже повільні. Так як необхідно використовувати велику кількість зображень для тренування нейронної мережі, це займе багато часу.

2.7 Порівняльна характеристика згорткових нейронних мереж

Для порівняння візьмемо Faster R-CNN, R-FCN, SSD, FPN, RetinaNet та YOLOv3 нейронні мережі і проведемо аналіз.

Для реальних додатків намагаються досягти балансу між точністю та швидкістю. Крім типів детекторів, нам потрібно знати про інші варіанти, які впливають на продуктивність:

- екстрактори функцій (VGG16, ResNet, Inception, MobileNet);
- вихідні кроки для екстрактора;
- дозвіл вхідного зображення;
- стратегія відповідності і поріг IoU (як виключаються прогнози при розрахунку втрат);
- поріг мінімального придушення IoU;
- жорсткий приблизний коефіцієнт видобутку (позитивне чи негативне ставлення якоря);
- кількість пропозицій або прогнозів;
- кодування границь вікна;
- збільшення даних;
- навчальний набір даних;
- використання багатомасштабних зображень при навчанні або тестуванні (з обрізанням);
- який шар(и) карти властивостей для виявлення об'єкта.
- функція втрати локалізації;
- використовується платформа для глибинного навчання;
- конфігурації навчання, включаючи розмір партії, зміна розміру вхідного зображення, швидкість навчання і розпад швидкості навчання.

Розглянемо результати роботи нейронних мереж на різних наборах даних.

Faster R-CNN, створений Шаоцін Рен (також співавтором якого був Гіршік), є третьою итерацією серії R-CNN. У Faster R-CNN додали мережу регіональних пропозицій (RPN), намагаючись позбутися від алгоритму вибіркового пошуку і зробити модель повністю навченою від кінця до кінця. Завдання полягає в тому, щоб виводити об'єкти на основі оцінки

«об'єктивності». Ці об'єкти використовуються пулом RoI і повністю підключеними шарами для класифікації.

На рисунку 2.4 представлені результати для тестового набору даних PASCAL VOC 2012. Розглянемо останні 3 рядки, що представляють продуктивність Faster R-CNN. У другому стовпці зазначено кількість RoI-ів, створених мережею пропозицій регіону. Третя колонка являє собою набір навчальних матеріалів. Четверта колонка - середнє значення (mAP) при вимірі точності.

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Рисунок 2.11 - VOC 2012 для Faster R-CNN

Результати на наборі даних MS COCO**.**

			COCO val		COCO test-dev	
method	proposals	training data	mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

Рисунок 2.12 - COCO для Faster R-CNN

Результати R-FCN на наборі даних PASCAL VOC 2012 (для деяких результатів використовується багатомасштабне навчання і тестування).

R-FCN - це фреймворк виявлення об'єктів на основі регіонів, яка використовує глибинні повністю згорткові мережі, які є точними і ефективними.

	training data	mAP (%)	test time (sec/img)
Faster R-CNN [9]	07++12	73.8	0.42
Faster R-CNN +++ [9]	07++12+COCO	**83.8**	3.36
R-FCN multi-sc train	07++12	77.6†	0.17
R-FCN multi-sc train	07++12+COCO	**82.0**‡	0.17

Рисунок 2.13 - VOC 2012 для R-FCN

Результати на наборі даних MS COCO.

	training data	test data	AP@0.5	AP	AP small	AP medium	AP large	test time (sec/img)
Faster R-CNN [9]	train	val	48.4	27.2	6.6	28.6	45.0	0.42
R-FCN	train	val	48.9	27.6	8.9	30.5	42.0	0.17
R-FCN multi-sc train	train	val	49.1	27.8	8.8	30.8	42.2	0.17
Faster R-CNN +++ [9]	trainval	test-dev	**55.7**	**34.9**	15.6	38.7	50.9	3.36
R-FCN	trainval	test-dev	51.5	29.2	10.3	32.4	43.3	0.17
R-FCN multi-sc train	trainval	test-dev	51.9	29.9	10.8	32.8	45.0	0.17
R-FCN multi-sc train, test	trainval	test-dev	**53.2**	**31.5**	14.3	35.5	44.2	1.00

Рисунок 2.14 - COCO для R-FCN

Результати SSD на PASCAL VOC 2007, 2012 і MS COCO наборах даних з використанням вхідних зображень 300 × 300 і 512 × 512 пікселів.

Single Shot Detector (SSD) забезпечує хороший баланс між швидкістю і точністю. SSD запускає згорткову мережу на вхідному зображенні тільки один раз і обчислює карту об'єктів.

Method	VOC2007 test		VOC2012 test		COCO test-dev2015 trainval35k		
	07+12	07+12+COCO	07++12	07++12+COCO			
	0.5	0.5	0.5	0.5	0.5:0.95	0.5	0.75
SSD300	74.3	79.6	72.4	77.5	23.2	41.2	23.4
SSD512	76.8	81.6	74.9	80.0	26.8	46.5	27.8
SSD300*	77.2	81.2	75.8	79.3	25.1	43.1	25.8
SSD512*	**79.8**	**83.2**	**78.5**	**82.2**	**28.8**	**48.5**	**30.3**

Рисунок 2.15 - Результати роботи SSD

SSD300 * і SSD512 * застосовують збільшення даних для невеликих об'єктів покращуючи mAP.

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Рисунок 2.16 - Результати роботи різних типів нейронних мереж

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

Рисунок 2.17 - Результати роботи SSD на наборі даних MS COCO

Розглянемо результати роботи YOLO нейронної мережі на наборах даних.

YOLO (You only Look Once) застосовує єдину нейронну мережу до повного зображенню. Ця мережа ділить зображення на регіони і передбачає обмежуючі поля і ймовірності для кожного регіону. Ці обмежуючі поля зважені за прогнозованими можливостям.

Результат представлений для набору даних VOC 2007, так як він містить результати для картинок з різним розширенням.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Рисунок 2.18 - VOC 2007 для YOLOv2

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

Рисунок 2.19 - Результат роботи YOLOv2 для набору даних PASCAL VOC 2012

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Рисунок 2.20 -Результат роботи YOLO для MS COCO

Результати роботи YOLOv3 на різних наборах даних.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Рисунок 2.21 - YOLOv3 для MS COCO

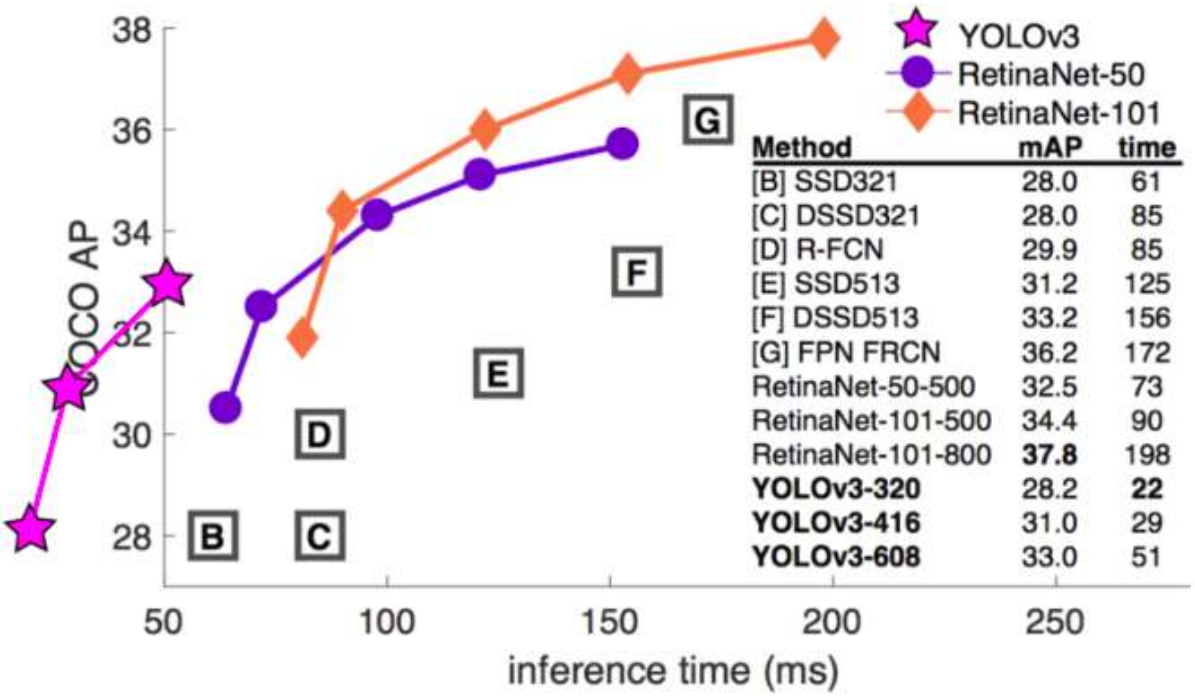

Рисунок 2.22 - Продуктивність для YOLOм3 з MS COCO

Результат роботи FPN нейронної мережі для MS COCO.

FPN (Feature Pyramid Networks) є основним компонентом в системах розпізнавання для виявлення об'єктів в різних масштабах. Але недавні детектори об'єктів глибинного навчання не використовували представлення про піраміди, тому що вони обчислюються і запам'ятовуються. Для побудови високорівневих карт семантичних функцій у всіх масштабах розроблена архітектура зверху вниз з бічними

з'єднаннями. Ця архітектура показує значне поліпшення як універсальний екстрактор функцій.

Faster R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP_s	AP_m	AP_l
(*) baseline from He *et al.* [16]†	RPN, C_4	C_4	conv5			47.3	26.3	-	-	-
(a) baseline on conv4	RPN, C_4	C_4	conv5			53.1	31.6	13.2	35.6	**47.1**
(b) baseline on conv5	RPN, C_5	C_5	2*fc*			51.7	28.0	9.6	31.9	43.1
(c) **FPN**	RPN, $\{P_k\}$	$\{P_k\}$	2*fc*	✓	✓	**56.9**	**33.9**	**17.8**	**37.7**	45.8

Рисунок 2.23 - Результат роботи FPN нейронної мережі для MS COCO

Результат роботи RetinaNet нейронної мережі для MS COCO.

RetinaNet - це єдина уніфікована мережа, що складається з магістральної мережі і двох підмереж, призначених для конкретних завдань. Основна відповідає за обчислення карти conv-функцій по всьому вхідному зображенню і є автономною мережею згортки. Перша підмережа виконує класифікацію на виході магістралей; друга підмережа виконує регресію прямокутника.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Рисунок 2.24 - Результат роботи RetinaNet нейронної мережі для MS COCO

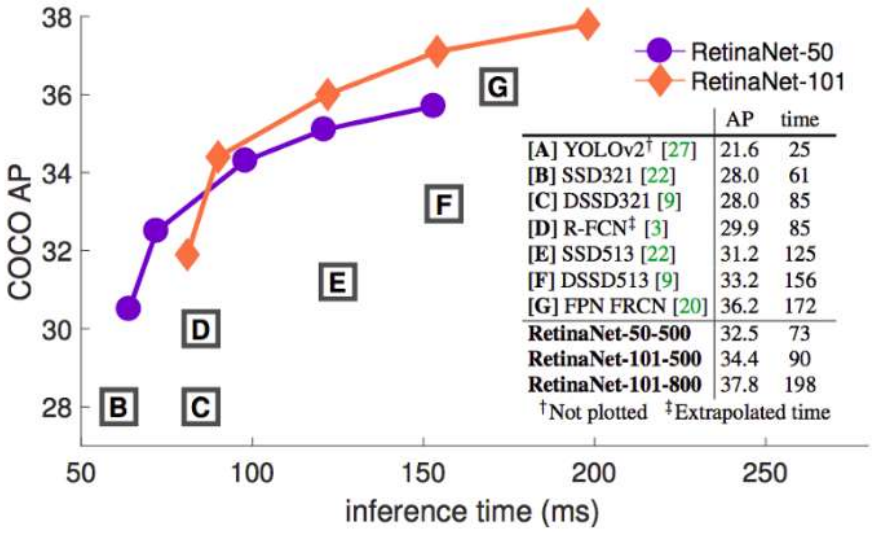

Рисунок 2.25 - Швидкість навчання в порівнянні з точністю для RetinaNet на MS COCO

Проаналізувавши отримані результати, зробили висновок, що найбільш релевантним буде використання Faster R-CNN нейронну мережу. Так як вона не вимагає великих технічних ресурсів для тренування, тренується досить швидко і при цьому показує прийнятний результат.

2.8 Принцип роботи Faster R-CNN

2.8.1 Поява Faster R-CNN

Faster R-CNN була спочатку опублікована в NIPS 2015. Після публікації вона зазнала кілька виправлень. Faster R-CNN є третьою итерацією документів R-CNN, в якій Росс Гіршік був автором і співавтором.

Початком можна вважати появу «Багатих ієрархій функцій для точного виявлення об'єктів і семантичної сегментації» (R-CNN) в 2014 році, в яких використовувався алгоритм «Селективний пошук» для визначення можливих областей інтересу і стандартна згорткова нейронна мережа (CNN) для їх класифікації та коригування. Алгоритм швидко перетворився в Fast R-CNN, опублікованому на початку 2015 року, де

технологія, яка називається Region of Interest Pooling, дозволяла ділитися дорогими обчисленнями і робила модель набагато швидше. Після цього була розроблена Faster R-CNN, де була запропонована перша повністю диференційована модель.

2.8.2 Архітектура Faster R-CNN

Архітектура Faster R-CNN комплексна, оскільки має кілька рухомих частин. Вона складається з базової нейронної мережі (CNN) і RPN.

Початком всього процесу можна вважати зображення, з якого необхідно отримати:

- список обмежують прямокутників;
- анотацію (лейбл), призначену кожному обмежує прямокутника;
- ймовірність для кожної анотації і обмежує рамки;

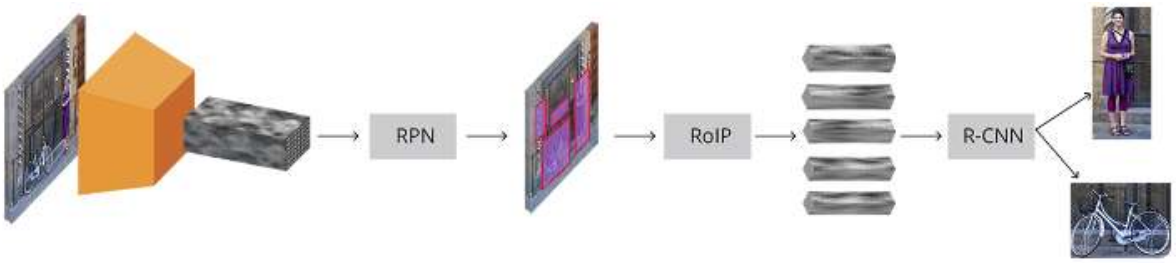

Рисунок 2.26 - Повна архітектура Faster R-CNN

Вхідні зображення представлені у вигляді Довжина × Ширина × Глибина тензора (багатовимірні масиви), які проходять через попередньо навчений CNN до проміжного рівня, який закінчується згортковим відображенням ознак. Це використовують як засіб виділення функцій для наступної частини.

Потім використовують RPN. Використовуючи функції, які CNN обчислює, RPN використовується для пошуку зумовленої кількості областей (обмежуючих полів), які можуть містити об'єкти.

Найбільш складна проблема з використанням Deep Learning (DL) для розпізнавання об'єктів - це створення списку обмежуючих прямокутників змінної довжини. При моделюванні глибинних нейронних мереж останній блок зазвичай являє собою тензорний розмір фіксованого розміру (за винятком випадків використання повторюваних нейронних мереж). Наприклад, при класифікації зображень вихід є N-подібний тензор, причому N представляє собою число класів, де кожен скаляр в місці розташування і містить ймовірність того, що це зображення є $lable_i$.

Завдання змінної довжини вирішується в RPN за допомогою якорів: фіксованих розмірних обмежують прямокутників, які розміщується рівномірно по всьому вихідного зображення. Замість того, щоб виявляти, де знаходяться об'єкти, розбивають проблему на дві частини. Для кожного якоря є два питання:

1. Цей якір містить відповідний об'єкт?
2. Як би ми відрегулювали цей якір, щоб краще відповідати відповідному об'єкту?

Після отримання списку можливих релевантних об'єктів і їх розташування в оригінальному документі, він стане простою проблемою, яку необхідно вирішити. Використовуючи функції, витягнуті CNN і обмежуючи поля відповідними об'єктами, застосовують Region of Interest (RoI) Pooling і витягують ті функції, які відповідали б відповідним об'єктам, в новий тензор.

Після цього підключають модуль R-CNN, який використовує цю інформацію для:

– класифікації вмісту в обмежуючій рамці;
– регулювання координати обмежуючої рамки (щоб він краще відповідав об'єкту).

2.8.3 Базова нейронна мережа

Першим кроком є використання CNN, попередньо обробленого для завдання класифікації (наприклад, з використанням ImageNet) і використання результату отриманого від проміжного шару.

Оригінальний Faster R-CNN використовував ZF і VGG нейронні мережі, попередньо оброблені на ImageNet, але з тих пір було багато різних мереж з різною кількістю ваг. Наприклад, MobileNet, менша і ефективна мережева архітектура, оптимізована для швидкості, має приблизно 3.3M параметрів, тоді як ResNet-152 (152 шару) має близько 60 мільйонів. Зовсім недавно нові архітектури, такі як DenseNet, покращують результати і знижують кількість параметрів.

2.8.4 VGG нейронна мережа

Розглянемо приклад роботи VGG-16 архітектури.

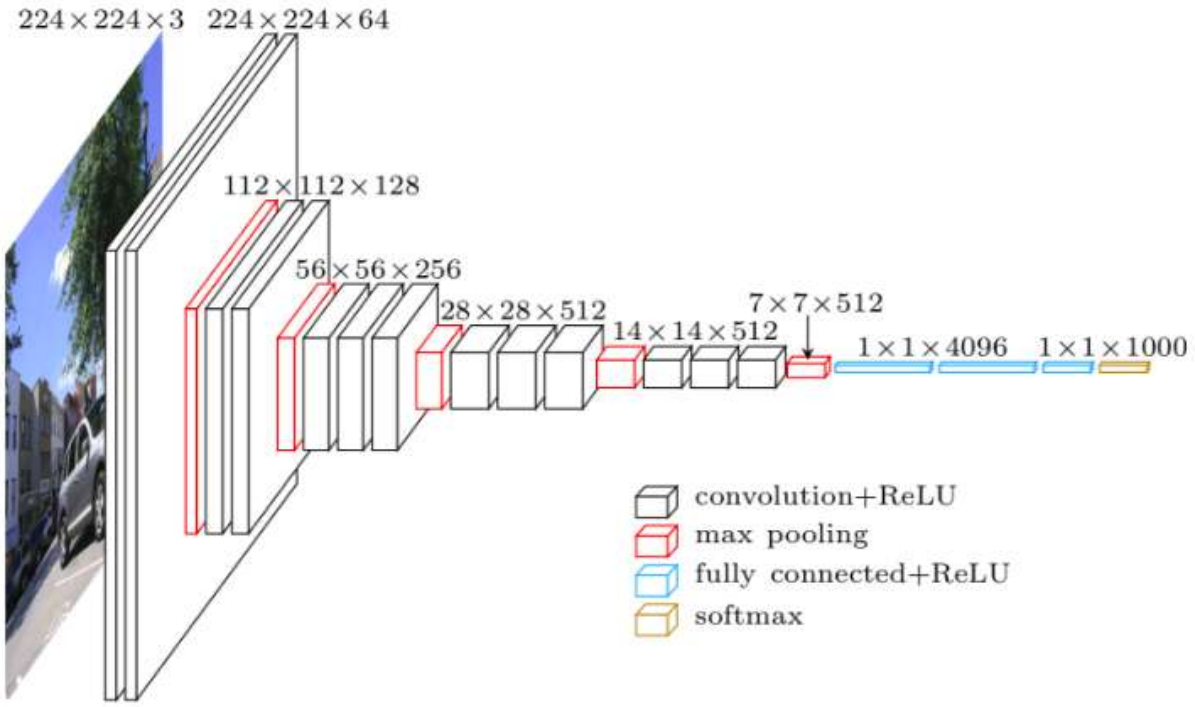

Рисунок 2.27 - VGG архітектура

VGG, назва належить команді, яка використовувала її в конкурсі ImageNet ILSVRC 2014 року, була опублікована в статті «Дуже глибинні

згорткові мережі для великомасштабного розпізнавання зображень» Карена Симоняна і Ендрю Зіссермана. За сьогоднішніми мірками вона не буде вважатися глибинною, але в той час вона більш ніж подвоїла кількість зазвичай використовуваних шарів (коли навчання можливо).

При використанні VGG для класифікації вхід являє собою 224 × 224 × 3 тензора (це означає зображення RGB розміром 224 на 224 пікселів). Він повинен залишатися фіксованим для класифікації, оскільки в останньому блоці мережі використовуються повністю пов'язані (FC) шари (замість згортальних), для яких потрібно вхід фіксованої довжини. Зазвичай це робиться шляхом згладжування вихідного сигналу останнього сверточного шару, отримання тензора рангу 1, перш ніж використовувати шари FC.

Так як слід використовувати результати отримані на проміжному згорточному шарі, розмір вхідних даних не є нашою проблемою. Принаймні, це не проблема цього модуля, оскільки використовуються тільки згорткові шари. В офіційній реалізації використовують результати рівня conv5 / conv5_1.

Кожен згортковий шар створює абстракції на основі попередньої інформації. Перші шари зазвичай вивчають ребра, другі знаходять шаблони по краях, щоб включити їх в більш складні форми і т. д. В кінці кінців ми отримаємо згорткову карту характеристик, яка має просторові розміри набагато менше вихідного зображення, але більшу глибину. Ширина і висота карти об'єктів зменшуються через об'єднання між згортковими шарами і збільшенням глибини в залежності від кількості фільтрів, які вивчає згортковий шар.

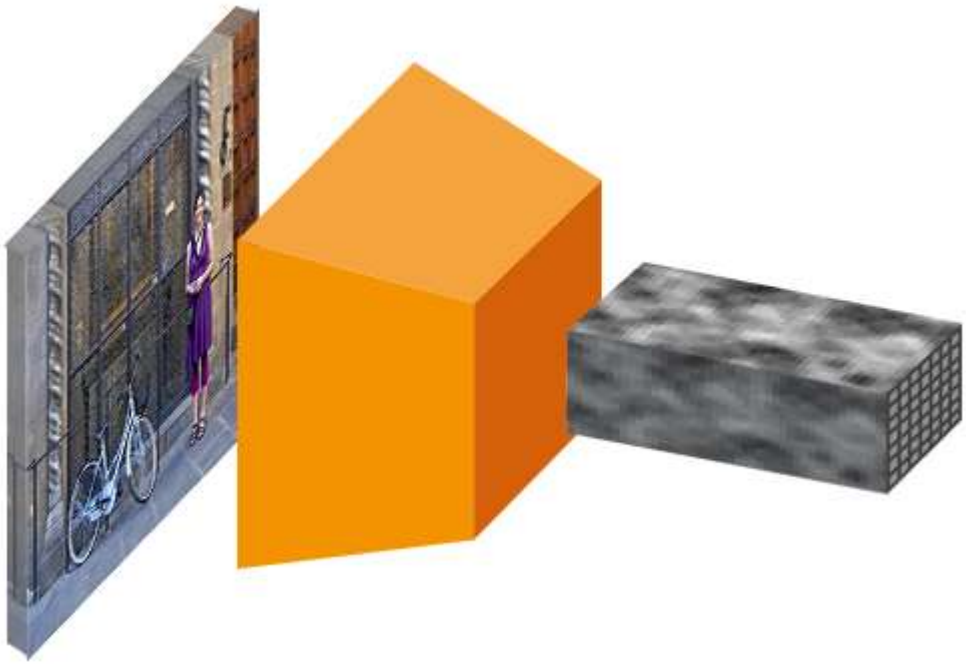

Рисунок 2.28 - Згортковий набір ознак (feature map)

Згортковий набір ознак кодував всю інформацію для зображення, зберігаючи при цьому місце розташування «речей», які він закодував щодо вихідного зображення. Наприклад, якщо в лівому верхньому кутку зображення з'явився червоний квадрат, і для нього активуються згорткові шари, тоді інформація для цього червоного квадрата буде як і раніше перебувати в верхньому лівому кутку згортальних наборів ознак.

2.8.5 Порівняння VGG і ResNet архітектур

На даний час архітектури ResNet в основному замінюють VGG як базову мережу для вилучення функцій. Три із співавторів Faster R-CNN (Kaiming He, Shaoqing Ren і Jian Sun) також були співавторами «Deep Residual Learning for Image Recognition», оригінальної статті, яка описує ResNets.

Очевидною перевагою ResNet над VGG є те, що він більше, тому він має більшу здатність «дізнатися», що потрібно. Це справедливо для завдання класифікації і має бути однаково справедливо в разі виявлення об'єкта.

Крім того, ResNet спрощує навчання глибинним моделям з використанням залишкових сполук і нормалізації партії, які не були винайдені, коли VGG був вперше випущений.

2.8.6 Якорі

Результатом роботи базової нейронної мережі є оброблене зображення, в якому необхідно знайти регіони, що представляють інтерес для класифікації. Якорі - це спосіб вирішити проблему зі змінною довжиною.

Мета - знайти обмежуючі прямокутники в зображенні. Вони можуть мати різні розміри і пропорції. Уявіть, що ми намагалися вирішити проблему, знаючи заздалегідь, що на зображенні є два об'єкти. Першою ідеєю, яка приходить на розум, є навчання мережі, яка повертає 8 значень: 2 множини x_{min}, y_{min}, x_{max}, y_{max}обмежують області для кожного об'єкта. Цей підхід має деякі фундаментальні проблеми. Наприклад, зображення можуть мати різні розміри і пропорції; наявність оптимальної моделі, призначеної для прогнозування вихідних координат, може виявитися дуже складним (якщо не неможливим). Іншою проблемою є неправильні передбачення: при прогнозуванні x_{min}, і x_{max}, необхідно забезпечити, щоб x_{min}, $<x_{max}$.

Існує більш простий підхід до прогнозування обмежуючих областей, шляхом навчання передбачення зсувів з посилальних блоків. Візьмемо контрольний блок x_{ctnter}, y_{center}, довжина, ширина і навчимо передбачати Δx_{center},Δy_{center}, $\Delta width$,$\Delta height$, які зазвичай є невеликими значеннями, які коригують контрольний блок, щоб краще відповідати бажаного результату.

Якорями є фіксовані обмежуючі прямокутники різного розміру і співвідношення, розміщені по всьому зображенню, які будуть використовуватися для посилання на перше прогнозоване місце розташування об'єкта.

Так як, використовується згортковий набір ознак розмірності $conv_{width} \times conv_{height} \times conv_{depth}$, створюється набір якорів для кожної з точок в $conv_{width} \times conv_{height}$. Важливо розуміти, що навіть якщо посилання визначені на основі згорткового набору ознак, то вони посилаються на вихідне зображення.

Оскільки є тільки згорткові і об'єднані шари, розміри набору ознак будуть пропорційні розмірам вихідного зображення. Математично, якщо зображення було $w \times h$, набір функцій буде закінчуватися так: $w / r \times h / r$, де r називається коефіцієнтом піддискретизації. Якщо визначити один якір для кожного положення набору ознак в просторі, остаточне зображення буде містити купу якорів, розділених r-пікселями. У разі VGG, $r = 16$.

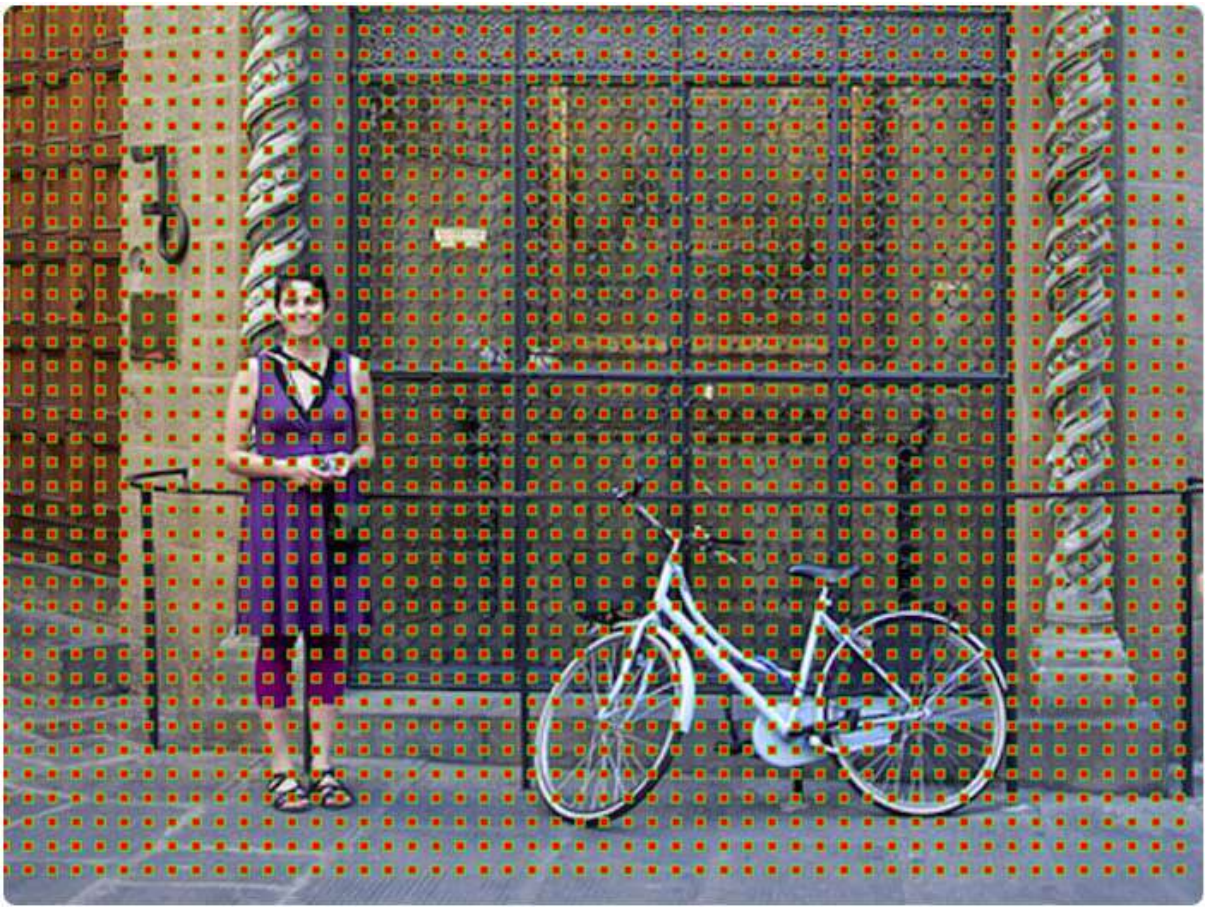

Рисунок 2.29 - Центри якорів по всьому вихідному зображенні

Вибір набору якорів визначається набором розмірів (наприклад, 64px, 128px, 256px) і набором коефіцієнтів між шириною і висотою

областей (наприклад, 0.5, 1, 1.5) і використовуються всі можливі комбінації розмірів і співвідношень сторін.

2.8.7 Принцип роботи RPN

RPN вживає всі посилальні області (якоря) і виводить набір хороших пропозицій для об'єктів. Він робить це, маючи два різних результати для кожного з якорів.

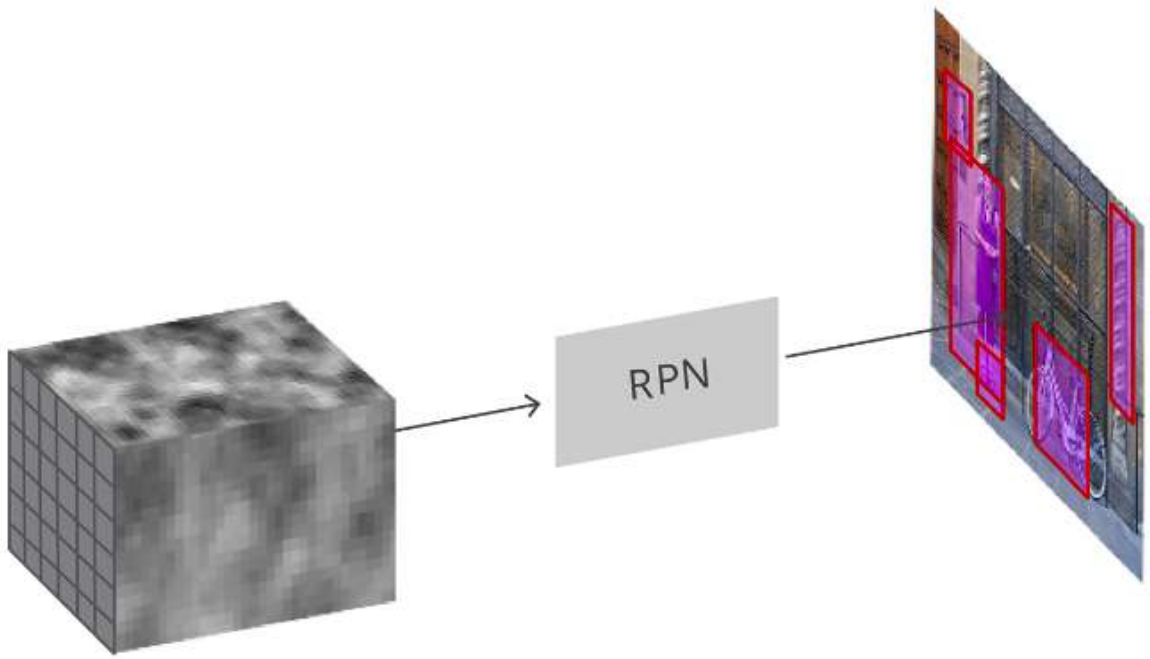

Рисунок 2.30 - RPN використовує згортковий набір ознак і генерує якоря на зображенні

Перший з них - ймовірність того, що якір є об'єктом («оцінка об'єктивності»). RPN не враховує, який клас об'єкта присутній, а тільки те, що він дійсно виглядає як об'єкт (а не фон). Цей показник об'єктивності використовують, щоб відфільтрувати погані прогнози для другого етапу. Другий результат - регресія прямокутника для налаштування якорів, щоб краще відповідати об'єкту, який він передбачає.

RPN ефективно реалізується повністю згортковим чином, використовуючи згортковий набір ознак, що повертається базовою

мережею в якості вхідних даних. По-перше, використовується згортковий шар з 512 каналами і розміром ядра 3 × 3, а потім два паралельних згортальних шари з використанням ядра 1 × 1, кількість каналів якого залежить від кількості якорів на кожну точку.

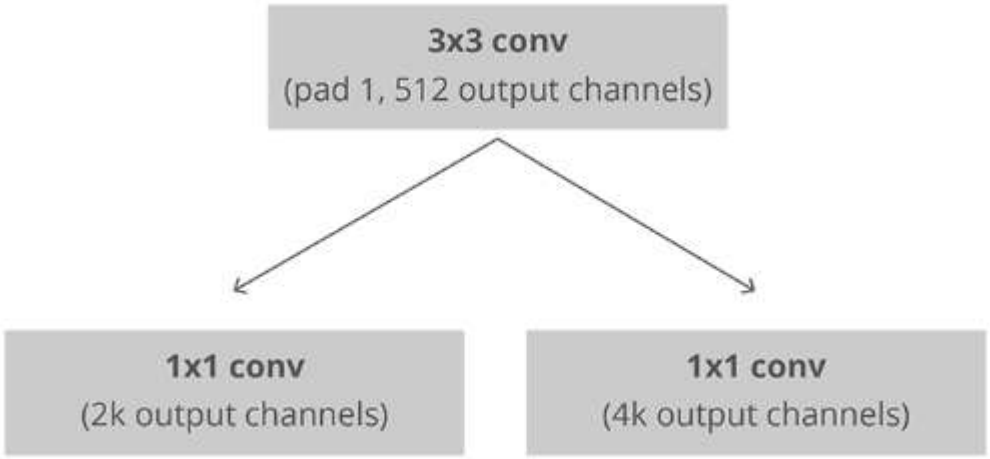

Рисунок 2.31 - Згорткова реалізація архітектури RPN, де k - кількість якорів

Для шару класифікації виводять два прогнози для кожного якоря: оцінка цього фону (а не об'єкта) і оцінка його переднього плану (фактичного об'єкта).

Для коригуючого шару регресії або прямокутника виводять 4 прогнозу: дельти Δx_{center},Δy_{center}, $\Delta width$,$\Delta height$, які будуть застосовуватися до якорів для отримання остаточних пропозицій.

Використовуючи остаточні координати пропозиції і їх оцінку «об'єктивності», отримують хороший набір пропозицій для об'єктів.

2.8.8 Об'єднання області інтересів

Після роботи RPN отримали безліч об'єктів пропозицій без присвоєного їм класу. Наступний крок полягає в тому, як взяти ці обмежують поля і класифікувати їх в потрібні категорії.

Найпростіший підхід - прийняти кожен пропозал (proposal), обрізати його, а потім пропустити його через попередньо підготовлену базову мережу. Потім використовувати витягнуті функції в якості вхідних даних для класичного класифікатора зображень. Основна проблема полягає в тому, що виконання обчислень для всіх пропозицій 2000 року неефективно і повільно.

Faster R-CNN вирішує цю проблему повторно використовуючи існуючий згортковий набір ознак. Це робиться шляхом вилучення набору ознак фіксованого розміру для кожної пропозиції з використанням об'єднання області інтересів. Для R-CNN необхідні карти функцій з фіксованим розміром, щоб класифікувати їх на фіксовану кількість класів.

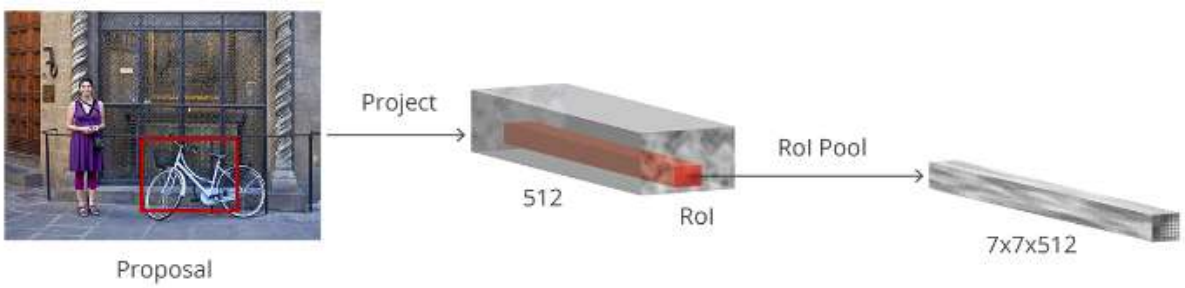

Рисунок 2.32 - Об'єднання області інтересів

Простіший метод, який широко використовується в реалізації виявлення об'єктів, в тому числі Luminoth's Faster R-CNN, полягає в тому, щоб обрізати згорткову карту об'єктів з використанням кожної пропозиції і потім змінити розмір кожної області до заданого розміру 14 × 14 × *convdepth* з використанням інтерполяції (зазвичай білінійної). Після обрізки, максимальний пул з ядром 2 × 2 використовується для отримання остаточної 7 × 7 × *convdepth* набору ознак для кожної пропозиції.

Причина вибору цих точних фігур пов'язана з тим, як вони використовуються таким блоком (R-CNN). Важливо розуміти, що вони налаштовуються в залежності від використання на другій стадії.

2.8.9 Згорткова нейронна мережа на основі регіонів

Згорткова нейронна мережа на основі регіонів (R-CNN) є заключним етапом в архітектурі Faster R-CNN. Отримавши з зображення згортковий набір ознак, використовуючи його для отримання пропозицій об'єктів за допомогою RPN і витягуючи функції для кожної з цих пропозицій (через RoI Pooling), використовувати ці функції для класифікації. R-CNN намагається імітувати заключні етапи класифікації CNN, де повністю підключений рівень використовується для виведення оцінки для кожного можливого класу об'єктів.

У R-CNN є дві різні цілі:

- класифікувати пропозиції в один з класів і клас фону (для видалення поганих пропозицій);
- кращого коригування обмежувальної рамки для пропозиції відповідно до прогнозованого класом.

В оригінальній Faster R-CNN документі, R-CNN відображає карту характеристик для кожної пропозиції, вирівнює її і використовує два повністю пов'язаних шари розміром 4096 з активацією ReLU.

Потім він використовує два різних повністю пов'язаних шару для кожного з різних об'єктів:

- повністю підключений рівень з $N + 1$ одиницями, де N - загальна кількість класів, і додатковий - для фонового класу;
- повністю підключений шар з блоками $4N$. Необхідно отримати передбачення для регресії, для цього використовують Δ_{center_x},Δ_{center_y}, Δ_{width},Δ_{height} для кожного з N можливих класів.

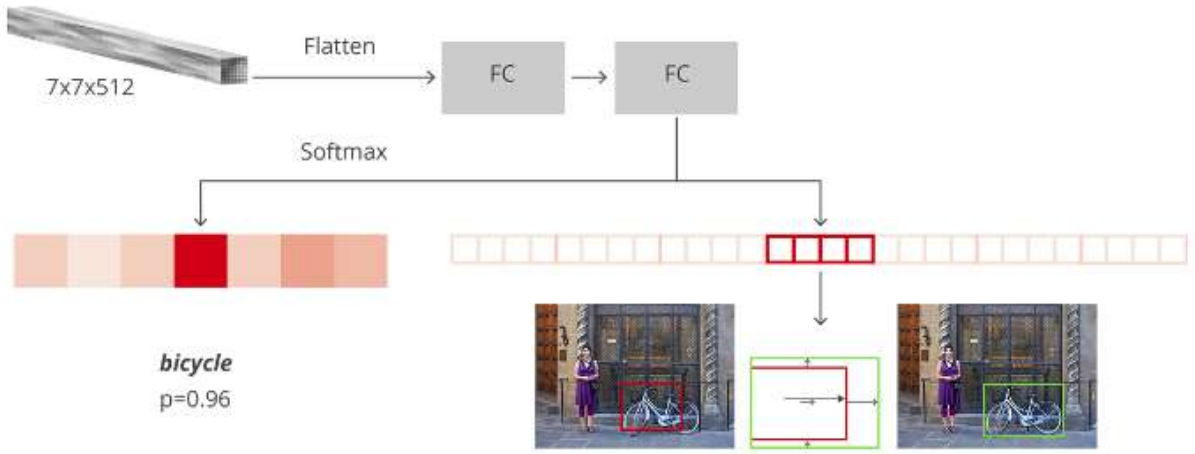

Рисунок 2.33 - Архітектура R-CNN

2.8.10 Оцінка

Оцінка виконується з використанням стандартної Mean Average Precision (mAP) на певному порозі IoU (наприклад, mAP 0.5). mAP - це метрика, яка відбувається з пошуку інформації і зазвичай використовується для обчислення помилки в завданнях ранжирування і для оцінки проблем виявлення об'єктів.

mAP «попереджає», коли пропустили поле, яке необхідно було виявити, а також коли виявляється щось, що не існує або виявляється один і той елемент кілька разів.

3 ОБГРУНТУВАННЯ МЕТОДІВ І ЗАСОБІВ РЕАЛІЗАЦІЇ

3.1 Мова програмування

Python - це мова загального призначення. Він має широкий спектр додатків з веб-розробки (наприклад, Django і Bottle), наукових і математичних обчислень (Orange, SymPy, NumPy) для настільних графічних користувальницьких інтерфейсів (Pygame, Panda3D).

У рейтингу мов програмування, які використовують для машинного навчання, Python займає лідируючі позиції: 57% вчених-розробників і machine learning розробників використовують цю мову програмування.

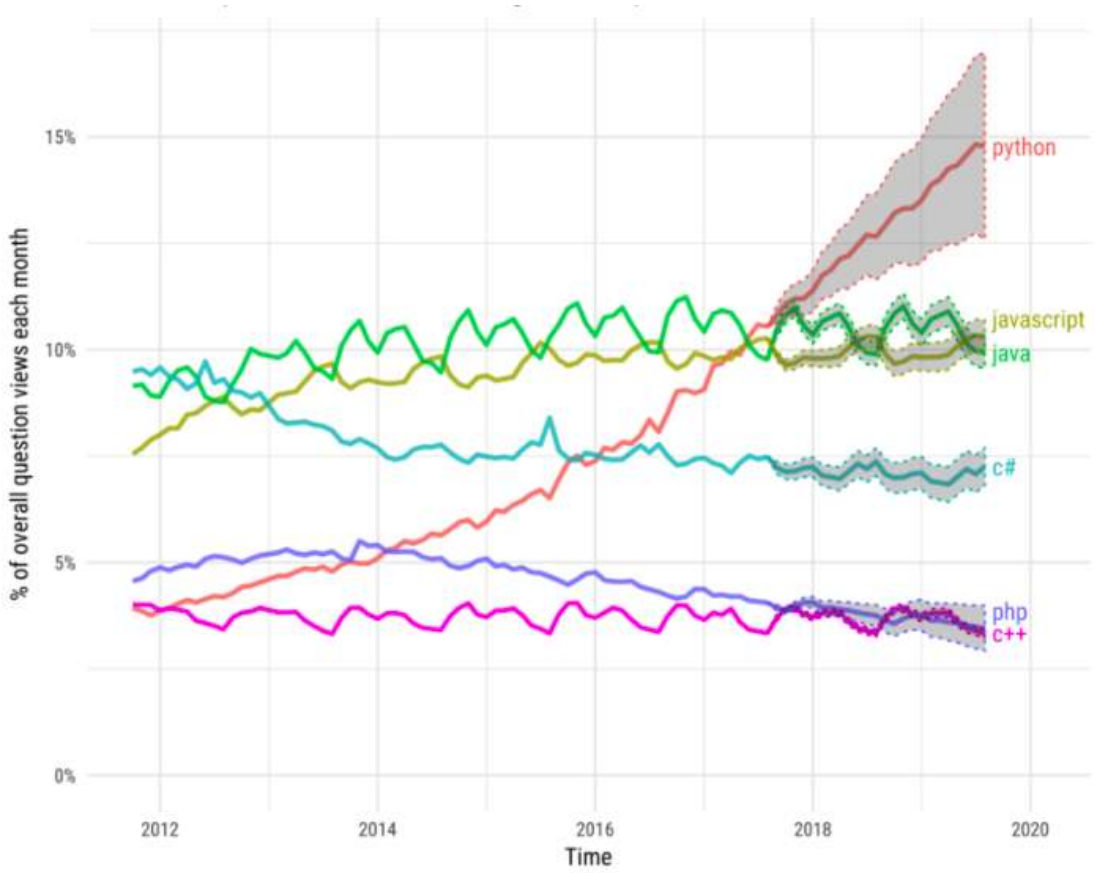

Рисунок 3.1 - Прогнози майбутнього трафіку для основних мов програмування

Рисунок 3.2 - Зростання популярності основних мов програмування

Розробники мови Python дотримуються певної філософії програмування, званої «The Zen of Python»:

- красиве краще, ніж потворне;
- явне краще, ніж неявне;
- просте краще, ніж складне;
- складне краще, ніж заплутане;
- пласке краще, ніж вкладене;
- розріджене краще, ніж щільне;
- читабельність має значення;
- особливі випадки не настільки особливі, щоб порушувати правила;
- при цьому практичність важливіше бездоганності;
- помилки ніколи не повинні замовчуватися;
- якщо не замовчуються явно;
- зустрівши двозначність, відкинь спокусу вгадати;

- повинен існувати один - і, бажано, тільки один - очевидний спосіб зробити це;
- хоча він спочатку може бути і не очевидний, якщо ви не голландець;
- зараз краще, ніж ніколи;
- хоча ніколи часто краще, ніж прямо зараз;
- якщо реалізацію складно пояснити - ідея погана;
- якщо реалізацію легко пояснити - ідея, можливо, хороша;
- простір імен - відмінна річ. Давайте будемо робити їх більше.

Мова програмування Python має ряд переваг:

- проста мова, яку легше вивчити: Python має дуже простий і елегантний синтаксис. Набагато простіше читати і писати програми на Python в порівнянні з іншими мовами, такими як: C ++, Java, C #. Python дозволяє зосередитися на вирішенні, а не на синтаксисі;
- вільний і відкритий навіть для комерційного використання. Є можливість не тільки використовувати і поширювати програмне забезпечення, написане на ньому, але і вносити зміни у вихідний код Python. Python має велике співтовариство, постійно вдосконалюється його на кожній ітерації;
- портативність: можливість переміщати програми Python з однієї платформи на іншу і запускати код без будь-яких змін. Він працює практично на всіх платформах, включаючи Windows, Mac OS X і Linux;
- розширюваний і вбудовуємий: якщо додаток вимагає високої продуктивності, то можливо легко комбінувати фрагменти C / C ++ або інших мов з кодом Python. Це надає додатком високу продуктивність, а також можливості сценаріїв, які інші мови не можуть надавати «з коробки»;
- високорівнева, інтерпретована мова: на відміну від C / C ++, не потрібно турбуватися про такі завдання, як управління пам'яттю, прибирання сміття та т. п. Аналогічно, коли запускається код Python,

він автоматично перетворює код в мову, який розуміє ваш комп'ютер , не потрібно турбуватися про будь-які операції нижчого рівня;

– великі стандартні бібліотеки для вирішення спільних завдань: Python має ряд стандартних бібліотек, що значно полегшує роботу програміста, так як не потрібно писати весь код самостійно. Наприклад: для підключення бази даних MySQL на веб-сервері використовують бібліотеку MySQLdb. Стандартні бібліотеки в Python добре протестовані і використовуються сотнями людей;
– об'єктно-орієнтованість: все в Python - це об'єкти. Об'єктно-орієнтоване програмування (ООП) допомагає вирішити складну проблему інтуїтивно. За допомогою ООП поділяють складні завдання на більш дрібні множини, створюючи об'єкти;

Існує ряд причин, за якими мова програмування Python популярна серед професіоналів, що працюють в системах машинного навчання.

Однією з найбільш причин є синтаксис Python, який був описаний як «елегантний», так і «математичний». Експерти відзначають, що семантика Python має особливу відповідність багатьом загальним математичним ідеям, так що для застосування цих математичних ідей на мові Python не потрібно витрачати багато часу на навчання новій мові програмування.

Python також часто описується як простий в освоєнні, що є його перевагою для будь-якого прикладного використання, включаючи системи машинного навчання. Деякі програмісти описують Python як сприятливий «компроміс складності і продуктивності» і описують, що використання Python більш інтуїтивно, ніж деякі інші мови, через його доступний синтаксис.

Інші користувачі відзначають, що Python також має спеціальні інструменти, які надзвичайно корисні при роботі з системами машинного навчання. Деякі приводять масив фреймворків і бібліотек, а також розширення, такі як NumPy, де ці інструменти спрощують реалізацію завдань на Python. Таким чином, контекст самого мови програмування також важливий в його популярності для застосування. Іншим ресурсом є

модуль scikit, званий «машинне навчання на Python», який може допомогти фахівцям у використанні Python.

Python описується краще для машинного навчання в порівнянні з такими мовами, як Java, Ruby on Rails, C або Perl. Там, де деякі можуть використовувати інші мови для «жорсткого кодування» і описувати Python як «іграшкова мова», доступний для основних користувачів, багато хто розглядає Python як повністю функціональну альтернативу використанню критичного синтаксису деяких інших мов.

Деякі відзначають, що простота використання спрощує спільне кодування і реалізацію, і що в якості мови загального призначення Python може легко виконувати багато функцій, що допомагає в складному наборі завдань машинного навчання. Все це робить Python популярною мовою програмування в технологічному світі. Іншою перевагою є широка підтримка: оскільки багато людей вважають Python стандартним, спільнота підтримки велика, що ще більше підвищує популярність Python.

3.2 Середовище розробки PyCharm

PyCharm забезпечує інтелектуальне завершення коду, перевірку коду, підкреслення помилок на льоту і швидкі виправлення, а також автоматичну реорганізацію коду і великі можливості навігації.

PyCharm надає відмінну підтримку для розробки з використанням сучасних веб-фреймворков, таких як Django, Flask, Google App Engine, Pyramid і web2py.

PyCharm інтегрується з IPython Notebook, має інтерактивну консоль Python і підтримує Anaconda, а також безліч наукових пакетів, включаючи matplotlib і NumPy.

На додаток до Python, PyCharm підтримує JavaScript, CoffeeScript, TypeScript, Cython, SQL, HTML / CSS, мови шаблонів, AngularJS, Node.js і інші мови програмування.

Також, дозволяє запускати, налагоджувати, тестувати і розгортати додатки на видалених хостингах або віртуальних машинах, з віддаленими інтерпретаторами, інтегрованим ssh-терміналом і інтеграцією Docker і Vagrant.

Колекція інструментів «з коробки»: інтегрований відладчик і тестовий інструментарій; Python профайлер; вбудований термінал; інтеграція з основними VCS і вбудованими інструментами бази даних.

3.3 Веб-фреймворк Django

Django - це високорівневий веб-фреймворк Python, який дозволяє швидко створювати безпечні і підтримувані веб-сайти. Django враховує безліч проблем веб-розробки, тому фреймворк надає інструментарій для комфортної розробки. Це безкоштовне і відкрите джерело, має процвітаючу і активну спільноту, відмінну документацію і безліч опцій для безкоштовної і платної підтримки.

Django дає можливість писати програмне забезпечення, яке є:

- повним;
- різностороннім;
- безпечним;
- масштабуємим;
- ремонтопригідним;
- портуємим;

Django слідує філософії «Батареї в комплекті» і надає майже все, що розробники можуть захотіти зробити «з коробки». Оскільки все є частиною одного «продукту», все це прекрасно працює разом, відповідає послідовним принципам розробки та має велику і сучасну документацію.

Django може бути використаний для створення практично будь-якого типу веб-сайту - від систем управління контентом і вікі, до соціальних мереж і новинних сайтів. Він може працювати з будь-якої

клієнтської платформою і може надавати контент практично в будь-якому форматі (включаючи HTML, RSS-канали, JSON, XML і т. Д.).

Django надає вибір практично будь-якої функції (наприклад, кілька популярних баз даних, шаблонізатор і т. Д.), Він також може бути розширений для використання інших компонентів, якщо це необхідно.

Django допомагає розробникам уникати багатьох поширених помилок безпеки, надаючи інфраструктуру, яка була розроблена для «правильної роботи», щоб автоматично захистити сайт. Наприклад, Django забезпечує безпечний спосіб управління обліковими записами користувачів і паролями, уникаючи поширених помилок, таких як включення інформації про сеанс в файли cookie, де вона вразлива (замість цього файли cookie містять тільки ключ, а фактичні дані зберігаються в базі даних) або безпосередньо зберігають паролі а не хеш пароля.

Хеш пароля - це значення фіксованої довжини, створене шляхом відправки пароля через криптографічну хеш-функцію. Django може перевірити правильність введеного пароля, виконавши його через хеш-функцію і порівнявши висновок зі збереженим значенням хеша. Однак через «односторонній» характер функції, навіть якщо збережене хеш-значення скомпрометовано, зловмисникові складно розробити вихідний пароль.

Django забезпечує захист від багатьох вразливостей за замовчуванням, включаючи SQL-ін'єкцію, міжсайтовий скриптинг, підробку і clickjacking.

Django використовує компонентну архітектуру «shared-nothing» (кожна частина архітектури не залежить від інших і тому може бути замінена або змінена, якщо необхідно). Чіткий поділ між різними частинами означає, що воно може масштабироваться для збільшення трафіку шляхом додавання обладнання на будь-якому рівні: кешування серверів, серверів баз даних або серверів додатків. Деякі з найбільш завантажених сайтів успішно масштабують Django для задоволення своїх вимог (наприклад, Instagram і Disqus).

Код Django написаний з використанням принципів і шаблонів дизайну, які заохочують створення підтримуваного і багаторазового коду. Зокрема, він використовує принцип Do not Repeat Yourself (DRY), тому немає непотрібного дублювання, що зменшує кількість коду. Django також сприяє угрупованню пов'язаних функцій в багаторазові «додатки» і на більш низькому рівні групи, пов'язані з кодом в модулі (відповідно до шаблону Model View Controller (MVC)).

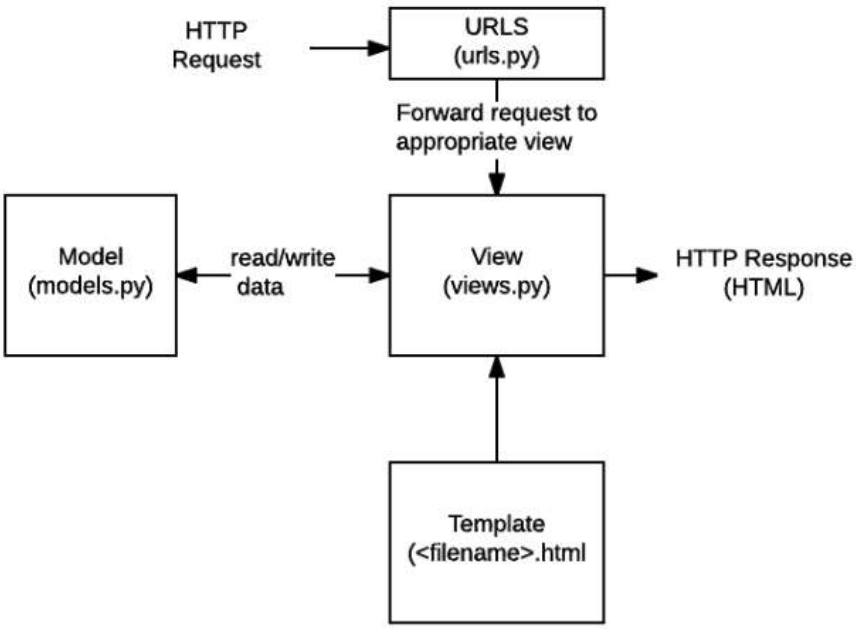

Рисунок 3.3 - Шаблон Model View Template (MVT)

Django написаний на Python, який працює на багатьох платформах. Це означає, що ви не прив'язані до якої-небудь конкретної серверної платформи і можете запускати свої додатки в багатьох варіантах Linux, Windows і Mac OS X. Крім того, Django добре підтримується багатьма постачальниками веб-хостингу, які часто надають певну інфраструктуру і документація для розміщення сайтів Django.

3.4 Бібліотека Tensorflow

TensorFlow - це бібліотека програмного забезпечення з відкритим вихідним кодом для чисельного розрахунку з використанням графіків потоку даних. Він був спочатку розроблений командою Google Brain в

рамках дослідницької організації Google Machine Intelligence для машинного навчання і досліджень глибоких нейронних мереж, але ця система є досить загальною, щоб бути придатною і в самих різних областях.

TensorFlow є крос-платформним. Він працює практично на всіх: графічних процесорах, включаючи мобільні та вбудовані платформи, і навіть вузли обробки тензорів (TPU), які є спеціалізованими апаратними засобами для створення тензорної математики (ще не доступні).

Для даного сервісу використовували TensorFlow Object Detection API - основне завдання машинного навчання для створення точних моделей машинного навчання, здатних локалізувати та ідентифікувати кілька об'єктів в одному зображенні. Нещодавно відкритий API TensorFlow Object Detection API розробив найсучасніші результати (і поміщений першим в задачу виявлення COCO).

Як модель вибрали faster_rcnn_inception_v2_coco і перетренували її.

4 ВИМОГИ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

4.1 Вимоги до системи в цілому

4.1.1 Вимоги до структури та функціонування системи

Система буде складатися з трьох частин:

- серверна частина;
- частина призначена для користувача;
- нейронна мережа.

Призначена для користувача частина повинна бути реалізована просто і бути інтуїтивно зрозумілою. Для цього необхідно підібрати правильні кольори, розміщення кнопок і їх підписи.

При розробці дизайну необхідно ознайомиться з основами Material Design. Ідея дизайну полягає в додатках, які відкриваються і згортаються як картки, використовуючи ефекти тіней. Переходи між елементами сторінки повинні бути плавними. Також, за можливості, не допускати нагромадження об'єктів на одній сторінці.

При створенні серверної частини необхідно дотримуватися одне з правил програмування на мові Python: «Do not Repeat Yourself». Тому, заздалегідь продумати які класи будуть створені і які методи їм будуть належати.

Все це впливає на швидкість роботи сервера і системи в цілому, що зменшить час очікування відповіді серверної частини на запит користувача.

Вимогою до нейронної мережі може служити те, що вона повинна працювати швидко, з низькими витратами ресурсів комп'ютера. Також, важливою складовою нейронної мережі є точність. Для цього необхідно правильно підібрати параметри моделі.

Так як дана нейронна мережа буде працювати з картинками, тому важливу роль займає якість і неоднорідність зображень, що так само необхідно враховувати при виборі параметрів моделі.

4.1.2 Вимоги до надійності

Веб-сервіс орієнтований на широкий ринок, тому необхідно враховувати одночасне відвідування сайту великою кількістю людей. А також забезпечити серверну частину механізмом обробки помилок, щоб виключити можливість припинення роботи сервера після виникнення помилки.

Щоб врахувати максимальну кількість варіантів можливих подій, необхідно проводити тестування системи і зберігати дані в лог-файли.

Система повинна бути стійка до великих навантажень і слабкого Інтернет-з'єднання. Ці аспекти необхідно враховувати при виборі платформи розгортання сервісу.

4.2 Вимоги до функцій, виконуваних системою

4.2.1 Підсистема збору, обробки, завантаження даних

Система розрахована на передачу, на прийом і повернення масивів даних: інформація користувача, масиви картинок.

Необхідно використовувати GET і POST запити при роботі з даними, що дозволить спростити роботу і швидкість відповіді сервера. Для отримання даних використовувати GET-запити, для оновлення даних в базі даних - POST-запити.

4.2.2 Підсистема зберігання даних

Зібрані дані від користувача необхідно зберігати в базі даних. На цьому етапі необхідно прийняти рішення, яка база даних буде використовуватися в системі.

При роботі з базами даних, необхідно використовувати CRUD функцію.

Необхідно додати індексування таблиць, для спрощення доступу до даних.

Залежно від використовуваної функції CRUD, буде використовуватися певний запит на сервер.

4.2.4 Підсистема роботи нейронної мережі

Нейронна мережа повинна бути правильно спроектована на початковому рівні розробки системи. Якість і швидкість роботи системи безпосередньо залежать від мережі.

Для досягнення цієї мети необхідно випробувати різні алгоритми нейронних мереж і вибрати найбільш підходящу модель.

Код, який буде написаний для взаємодії мережі і сервера повинен бути простим в розумінні і легко модифікується. Тому, необхідно робити правильні і корисні коментарі.

5 ОПИС МОДУЛІВ ПРОГРАМИ

5.1 Структура програми

Програма складається з декількох частин:

- нейронна мережа, натренована з використанням Python бібліотеки Tensorflow;
- серверна частина з використанням фреймворку Django;
- призначена для користувача частина - мова розмітки HTML, каскадна таблиця стилів CSS, мова програмування JS і запити, які реалізовані за використання AJAX.

Однією з переваг використання веб-фреймворку Django є те, що він надає можливість жорсткої структуризації проекту по папках, де кожна частина знаходиться в окремій директорії.

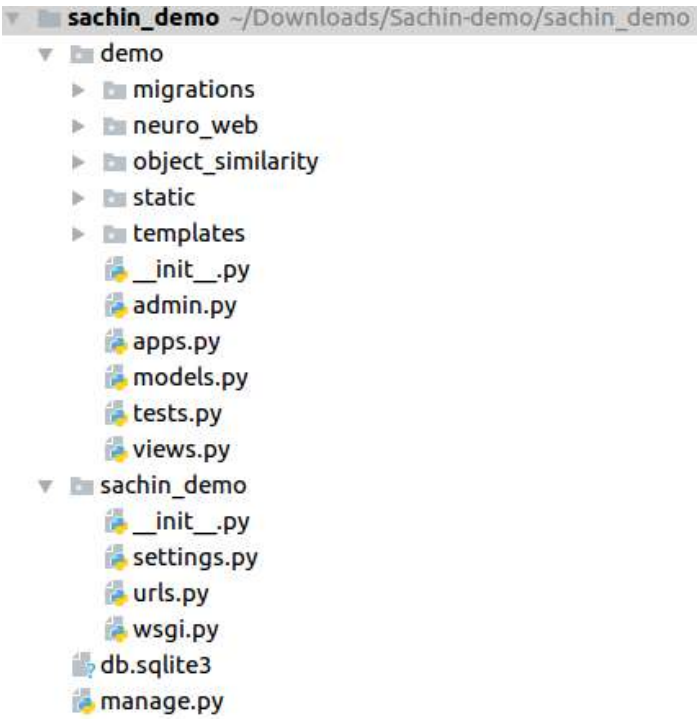

Рисунок 5.1 - Структура серверної частини

Папка «sachin_demo» є кореневою і містить в собі підпапки та файли:

- sachin_demo;

– demo;
– db.sqlite3;
– manage.py.

Підпапка «sachin_demo» містить в собі конфігураційні файли сервера.

У файлі «settings.py» можна прописати необхідні настройки сервера:

– локалізацію;
– часовой пояс;
– базу даних для використання;
– шляху, де знаходяться стилі, медіа-файли, шаблони.

Також, цей файл містить 2 важливі параметри. секретний ключ - використовується для кодування і захисту даних; режим роботи сервера: режим розробки або продакшн режим (в процесі створення сервера, режим повинен бути встановлений для розробки, після завершення - вказати продашн режим).

У файлі «urls.py» вказані основні юрл-и, які, якщо це необхідно, дозволяють отримати доступ до певного набору функцій (наприклад, для цього проекту, основний функціонал знаходиться в папці «demo»).

В папці «demo» знаходяться всі файли, які забезпечують взаємодію нейронної мережі, її результату і запиту користувача.

У файлі «views.py» є сполучною ланкою між запитом користувача і відповіддю нейронної мережі.

В папці «migrations» збережені всі зміни, внесені в базу даних.

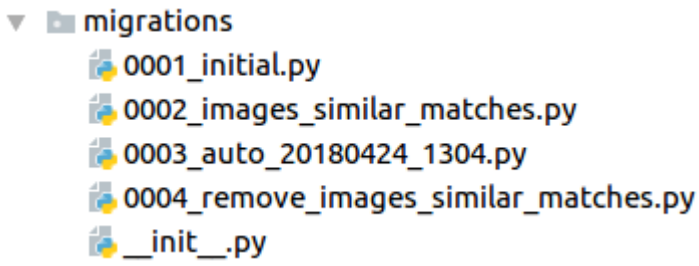

Рисунок 5.2 - Структура папки «migrations»

В папці «neuro_web» знаходяться всі папки та файли, пов'язані з нейронною мережею.

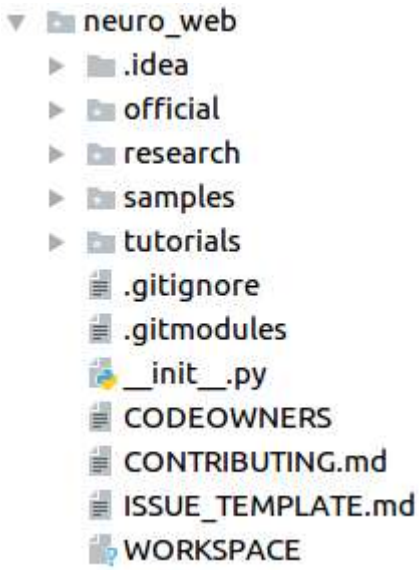

Рисунок 5.3 - Структура папки «neuro_web»

В папці «object_similarity» знаходиться алгоритм, який шукає схожі елементи між картинкою користувача і результатом, отриманим після роботи нейронної мережі, і сортує картинки за ступенем їх схожості.

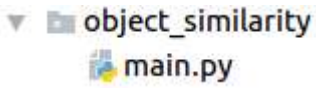

Рисунок 5.4 - Структура папки «object_similarity»

В папці «static» знаходяться всі файли стилів, js-файли і картинки, які повертаються як результат користувачеві.

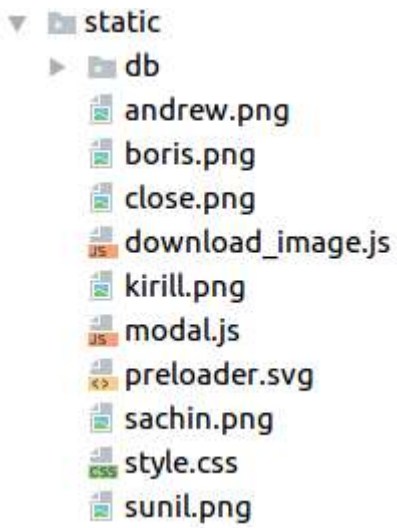

Рисунок 5.5 - Структура папки «static»

Папка «db» містить папки з картинками за категоріями одягу. На даний момент, нейронна мережа вміє розпізнавати 50 категорій одягу.

Папка «templates» містить шаблон сайту.

Рисунок 5.6 - Структура папки «templates»

5.2 Принцип роботи програми

Принцип роботи програми можна описати в кілька кроків:

1) користувач заходить на сайт;
2) користувач вибирає зображення, збережене на його пристрої;
3) відправляється POST-запит на сервер;
4) дані від користувача валідуються;
5) нейронна мережа намагається розпізнати, що за річ зображена на зображенні;
6) сервер приймає результат роботи нейронної мережі (тип одягу, який знаходиться на зображенні);
7) отримує масив картинок з бази даних;
8) сортує картинки по «схожості» з призначеної для користувача;

9) повертає результат користувачеві.

ВИСНОВКИ

У машинного навчання безліч шляхів розвитку. Вже зараз моделі машинного навчання можуть використовуватися для підвищення ефективності, виявлення ризиків або нових можливостей і застосування додатків в багатьох різних секторах. Вони або прогнозують точне значення (наприклад, продаж наступного тижня), або пророкують групи значень, наприклад, чи є клієнт високим ризиком, середнім ризиком або низьким ризиком.

Провівши дослідницьку роботу під час написання диплома та створення тестової версії програми, з'ясували, що даний проект має безліч шляхів застосування і є прикладом розвитку технологій на сьогоднішній день.

Отримані результати можна покращувати, використовуючи різноманітні підходи: змінювати модель, параметри нейронної мережі, використовувати іншу архітектуру нейронної мережі або додавати кількість даних під час тренування.

Розглянули принципи роботи нейронних мереж і їх математичну реалізацію. Також, вивчили принципи створення сервісів і представили тестовий варіант програми. Наступними етапами розвитку роботи є створення повноцінного сервісу, з реєстрацією та оплатою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Машинне навчання на Python, перше видання: навчальний посібник / Себастіан Рашка, Бірмінгем, 2015. 420с.

2. Штучний інтелект для людей випуск 3: Глибинне навчання і нейронні мережі: навчальний посібник / Джеф Хетон, Честрерфілд, 2015. 310 с.

3. Введення в математику нейронних мереж: навчальний посібник / Джеф Хетон, Честрерфілд, 2012. 270 с.

4. Пономаренко, Л. А. Построение оптимальной последовательности соединения отношений в запросах реляционной базы данных / Л. А. Пономаренко, С. С. Танянский, В. А. Филатов. // Системні дослідження та інформаційні технології. Міжнародний науково-технічний журнал. – 2003. – № 2. – С. 53-58.

5. Н.В. Касаткина, С.С. Танянский, В.А. Филатов Методы хранения и обработки нечетких данных в среде реляционных систем // Автоматика. Автоматизація. Електротехнічні комплекси та системи. – ХНТУ. – Херсон. – 2009. – випуск 2 (24) . – С.80-86.

6. Филатов В.А. Мультиагентные технологии интеграции гетерогенных информационных систем и распределенных баз данных : Дис. д-ра техн. наук: 05.13.06 / Харьковский нац. ун-т радиоэлектроники. – 2004. – 341с.

7. Касаткина, Н. В. Построение системы функциональных ассоциативных правил на основе свойств реляционной модели данных [Текст] / Н. В. Касаткина, Л. А. Пономаренко, С. С. Танянский, В. А. Филатов. // Збірник наукових праць Військового інституту КНУ ім. Т. Шевченка. – К. : ВІНКУ, 2009. – випуск 23. – С. 34-38.

8. Филатов В.А. Модель мультиагентной системы автономного администрирования информационных систем и распределенных баз данных / В. А. Филатов, Е. Е. Цыбульник, Л. Э. Чалая. // Новости искусственного интеллекта, 2002, № 4, с. 620-627.

9. В.А. Филатов, В.А. Кривоносов, О.Ф. Козырь Адаптивные автономные сценарии в задачах управления информационными ресурсами предприятия // Инженерный вестник Дона, 2013, № 3.

10. В.А. Филатов Модель поведения автономного агента на основе теории автоматов / В. А. Филатов // Вестник Херсонского государственного технического университета, Херсон. - 2004, № 1(19), С.108-111.

11. В.М. Левыкин, В.А. Филатов, Н.В. Черненко Метод синтеза единой структурной составляющей реляционной модели данных // УСиМ: Управляющие системы и машины, № 6, 2011. – С. 10-13.

12. В.А. Филатов, Е.Б. Чапланова Development of Information Technology of Object-relational Databases Design / Разработка информационной технологии проектирования объектно-реляционных баз данных // European Researcher, 2012, Vol.(36), № 12- P/ 2095-2101.

13. Filatov V. Fuzzy models presentation and realization by means of relational systems // Econtechmod : an international quarterly journal on economics in technology, new technologies and modelling processes. – Lublin ; Rzeszow, 2014. – Vol.(3), № 3. – P. 99-102.

14. Filatov V., Radchenko V. Reengineering relational database on analysis functional dependent attribute // Proceedings of the X Intern. Scient. and Techn. Conf. «Computer Science & Information Technologies» (CSIT’2015), 14-17 sept. 2015. – Lviv, Ukraine. – P. 85-88.

15. Filatov V., Voloshchuk O., Spivak N. Implementation and support fuzzy systems by means the relational data model // «Współpraca Europejska»/ «European Cooperation», Vol 4, No 11 (2016). – P. 49-61.

16. Нейронні мережі для розпізнавання образів / Крістофер М. Бішоп, Університет Оксфорд, Нью-Йорк, 2005. 200 с.

17. Дизайн нейронної мережі / Мартін Т. Хаган, Говард Б. Демут, Нью-Йорк, 2014. 386 с.

18. Майкл Нільсен / Нейронні мережі і Глибинне навчання, 2017, URL: http://neuralnetworksanddeeplearning.com/