**Методи класифікації зображень нейромережами**

Основні підходи класифікації зображень

Існує два основних підходи класифікації зображень: класифікація на основі пікселів та класифікація на основі об’єктів.

Пікселі є базовими елементами зображення і класифікація на основі аналізу пікселів є основним методом. Існують алгоритми, що або використовують тільки спектральну інформацію окремих пікселів для класифікації, або поєднують спектральну інформацію з просторовою (дані про прилеглі пікселі). Методи класифікації на рівні пікселів враховують лише яскравість кожного пікселя, тоді як об’єктна класифікація використовує як спектральну, так і просторову інформацію.

До методів класифікації на основі пікселів належать такі, як метод мінімальної відстані до середнього, максимальна ймовірність та метод відстані Махаланобіса. Вони використовують середні значення та дисперсії для різних класів і працюють за принципом вимірювання «відстані» між цими середніми значеннями та значеннями пікселів. Методи класифікації на основі пікселів мають обмеження, оскільки не враховують інформацію з сусідніх пікселів, але мають перевагу в простоті реалізації та ефективності у задачах з базовими візуальними характеристиками.

Натомість класифікація на рівні об’єктів враховує і спектральні, і просторові дані, оскільки включає прилеглі пікселі в процес класифікації. Під «об’єктом» розуміють не конкретний об’єкт на зображенні, а суміжну область пікселів. Цей підхід базується на розпізнаванні об’єктів або груп пікселів, які утворюють певні смислові одиниці на зображенні. Він дозволяє враховувати не тільки самі пікселі, але й їх розташування та взаємозв’язки між ними.

Таким чином, класифікація на основі пікселів більше підходить для простих задач, тоді як класифікація на основі об’єктів є більш універсальною та потужною для складних візуальних задач.

Попередня обробка даних

Сучасні та надійні системи класифікації зображень здебільшого використовують підхід, орієнтований на об’єкти, де зображення потрібно відповідним чином підготувати. Спочатку необхідно вибрати та попередньо обробити об’єкти або області на зображенні.

Перед тим як класифікувати зображення чи його окремі об’єкти, комп’ютер повинен «зрозуміти» дані, які воно містить. Це досягається шляхом попередньої обробки та підготовки зображень для подальшого введення в класифікаційний алгоритм, використовуючи техніки виявлення об’єктів. Цей процес є важливим етапом підготовки зображень і даних для навчання моделей машинного навчання.

Попередня обробка зображень є важливою для застосунків комп’ютерного зору, оскільки вона покращує точність до 30% [6]. Наступні етапи попередньої обробки відіграють дуже важливу роль в оптимізації зображень для аналізу, оскільки покращують вхідні дані для моделей машинного навчання:

– зменшення шуму;

– підвищення контрастності;

– зміна розміру;

– нормалізація;

– аугментація;

– перетворення кольорових просторів (наприклад, з RGB у відтінки сірого або HSV);

– виділення контурів;

– підвищення різкості;

– видалення небажаних артефактів.

Правильна попередня обробка зображень не тільки підвищує точність, але й знижує обчислювальну складність, полегшуючи моделі навчання важливим ознакам і покращуючи їх роботу в реальних додатках.

Локалізація зображення

Виявлення об’єктів здійснюється за допомогою різних методів. Спосіб попередньої обробки залежить від того, чи на зображенні присутній один об’єкт інтересу, чи їх кілька. Якщо на зображенні є лише один об’єкт, застосовується техніка локалізації зображення. Пікселі зображення мають числові значення, що інтерпретуються комп’ютером для відтворення відповідних кольорів і відтінків. Навколо об’єкта, що цікавить, створюється обмежувальна рамка, яка вказує комп’ютеру, яка частина зображення є важливою та які піксельні значення її описують.

Локалізація є важливим етапом у процесі розпізнавання об’єктів, оскільки вона допомагає не тільки визначити наявність об’єкта, але й точно вказати його місцезнаходження на зображенні. Обмежувальна рамка дозволяє моделі фокусуватися на конкретній частині зображення, що спрощує подальшу класифікацію (рис. 2.1). У багатьох випадках локалізація може бути доповнена додатковими параметрами, наприклад, оцінкою впевненості в тому, що об’єкт дійсно є в межах обмежувальної рамки. Ця інформація особливо важлива в складних або завантажених сценах, де можуть бути кілька схожих об’єктів або де частини об’єктів можуть бути частково закриті іншими елементами зображення.

Зображення, що містить ссавець, текст, колаж, собака

Автоматично згенерований опис

Рисунок 2.1 – локалізація зображення

Виявлення об’єктів

Якщо ж на зображенні декілька об’єктів, застосовується метод виявлення об’єктів (рис. 2.2), щоб визначити обмежувальні рамки для кожного об’єкта. Цей метод дозволяє одночасно ідентифікувати та локалізувати кілька об’єктів на одному зображенні, що є важливим аспектом у багатьох застосуваннях комп’ютерного зору. Виявлення об’єктів може виконуватися за допомогою різних алгоритмів, таких як YOLO, SSD та Faster R-CNN. Кожен з цих алгоритмів має свої переваги та недоліки, проте всі вони намагаються оптимізувати процес виявлення, щоб зменшити час обробки та підвищити точність. Наприклад, YOLO обробляє зображення за один прохід, що дозволяє досягти високої швидкості в реальному часі, тоді як Faster R-CNN використовує попередньо обчислені регіональні пропозиції, що забезпечує високу точність, але вимагає більше часу на обробку. Після виявлення об’єктів, для кожного з них створюються обмежувальні рамки, які вказують на їх точне розташування.

Зображення, що містить транспортний засіб, Наземний транспортний засіб, колесо, текст

Автоматично згенерований опис

Рисунок 2.2 – виявлення об’єктів на зображенні

Архітектура згорткових нейронних мереж

Класична архітектура CNN складається з різних шарів, кожен з яких виконує унікальні функції, необхідні для ефективного аналізу зображень. Ці шари взаємодіють для вилучення ознак, їх обробки і, зрештою, класифікації. Основні типи шарів у CNN:

– згортковий шар. Цей шар є основою нейронної мережі. У цьому шарі використовуються невеликі фільтри для сканування зображення і виявлення різних ознак, таких як краї, кути або прості геометричні фігури. Кожен фільтр згортається по зображенню, виконуючи операцію згортки, щоб створити карти ознак, які містять інформацію про специфічні патерни в зображенні. Це дозволяє мережі «бачити» локальні структури на зображенні і поступово виділяти більш складні ознаки на глибших рівнях мережі;

– пулінговий шар використовується для зменшення розмірності карт ознак, що отримані після згорткових шарів. Це зменшує кількість параметрів і обчислень у мережі, зберігаючи найбільш важливі ознаки;

– повнозв’язні шари розташовані в кінці архітектури нейронної мережі і відповідають за остаточну класифікацію. У повнозв’язних шарах кожен нейрон пов’язаний з усіма нейронами попереднього шару, що дозволяє мережі об’єднати всю вилучену інформацію з попередніх згорткових і пулінгових шарів. Основна функція цього шару – узагальнення виділених ознак і їх перетворення в ймовірності для кожного можливого класу. На виході останнього повнозв’язного шару зазвичай використовується функція Softmax, яка перетворює значення у ймовірності класів, забезпечуючи вибір найбільш ймовірного класу для зображення.

Класична архітектура CNN (рис. 2.3) складається з компонентів, що працюють разом для ефективного аналізу зображень. Згорткові шари виділяють ознаки, пулінгові шари зменшують розмірність даних, а повнозв’язні шари виконують остаточну класифікацію. Разом ці компоненти створюють потужну систему для вирішення завдань комп’ютерного зору.

Зображення, що містить текст, схема, ряд, Шрифт

Автоматично згенерований опис

Рисунок 2.3 – класична архітектура CNN

Процедура класифікації за допомогою CNN здійснюється за наступною схемою:

Крок 1. На вхід мережі подається зображення, яке представляється як тривимірний тензор (ширина, висота, кількість каналів). Наприклад, для кольорового зображення розміром 32×32 пікселів, вхід буде 32×32×3 (3 канали для кольорів RGB).

Крок 2. До вхідного зображення застосовується згортковий шар. У цьому шарі використовуються кілька фільтрів (наприклад, 5×5), які сканують зображення і виділяють локальні ознаки, такі як краї або текстури.

Крок 3. Кожен фільтр створює карту ознак, яка показує, де на зображенні присутні виділені фільтром ознаки. Зазвичай кілька фільтрів використовуються одночасно, що дозволяє отримати кілька карт ознак.

Крок 4. Застосовується пулінговий шар, зазвичай максимальний пулінг. Він зменшує розмір карт ознак, зберігаючи найважливішу інформацію. Наприклад, максимальний пулінг з вікном 2×2 вибирає максимальне значення в кожній області розміру 2×2 і створює зменшену карту ознак. Це зменшення розмірності допомагає знизити кількість параметрів і обчислень, підвищуючи ефективність мережі.

Крок 5. Згорткові та пулінгові шари зазвичай повторюються кілька разів для виділення більш складних і абстрактних ознак на різних рівнях. На кожному етапі глибина мережі збільшується, і мережа навчається розпізнавати складніші патерни.

Крок 6. Після кількох рівнів згорткових і пулінгових шарів, тривимірні карти ознак перетворюються у вектор (одномірний масив) за допомогою процесу, званого флатенінг. Це потрібно для підготовки даних до повнозв’язних шарів.

Крок 7. Вектор передається на вхід повнозв’язного шару, де кожен нейрон з’єднаний з усіма нейронами попереднього шару. Це дозволяє об’єднати інформацію з усіх ознак, виділених у попередніх шарах.

Крок 8. Повнозв’язні шари працюють на рівні класифікації і допомагають мережі передбачати належність зображення до одного з класів.

Крок 9. Останній шар мережі – це вихідний шар, який використовує функцію Softmax для багатокласової класифікації. Ця функція перетворює значення на виході повнозв’язного шару в ймовірності для кожного класу.

Крок 10. На основі ймовірностей мережа робить остаточне передбачення, визначаючи клас, до якого належить зображення.

Крок 11. Під час навчання мережа проходить через цикли прямого і зворотного поширення помилки. У прямому проході мережа обчислює передбачення, а в зворотному — коригує свої ваги, щоб зменшити різницю між передбаченими значеннями і реальними мітками зображень.

Крок 12. Після навчання мережа оцінюється на тестовій вибірці, щоб перевірити її здатність класифікувати нові, невідомі раніше зображення. Якщо точність передбачень недостатня, архітектуру мережі можна вдосконалити, додавши або змінюючи кількість згорткових шарів, використовуючи регуляризацію або інші методи.

Сучасні архітектури

Глибоке навчання стало ключовим підходом у класифікації зображень завдяки здатності CNN автоматично вивчати високорівневі ознаки зображень. Із розвитком обчислювальних потужностей та збільшенням кількості даних дослідники почали створювати більш складні архітектури CNN, які дозволяють покращити точність моделей та їх ефективність.

ResNet

ResNet (Residual Network) є однією з найвідоміших і найуспішніших архітектур у галузі глибокого навчання, яка зробила прорив у тренуванні дуже глибоких нейронних мереж. Вона була запропонована в 2015 році командою дослідників з Microsoft. Основною ідеєю ResNet є використання залишкових з’єднань, що дозволяють обійти проблеми тренування глибоких мереж, зокрема зникання градієнта.

У класичних глибоких нейронних мережах із великою кількістю шарів виникає проблема деградації градієнта під час навчання – коли мережа стає дуже глибокою, тренування може ставати менш ефективним, а точність навіть зменшується. Це відбувається через те, що градієнт, який передається назад під час зворотного поширення, може зникати або занадто зменшуватися з кожним шаром, що заважає ефективному оновленню ваг.

У класичних глибоких мережах збільшення кількості шарів може призвести до зниження точності через труднощі з оптимізацією, зокрема через проблему зникання або вибуху градієнтів. ResNet вирішує цю проблему за допомогою залишкових з’єднань (рис. 2.4), додаючи прямі шляхи передачі інформації від одного шару до іншого, оминаючи деякі проміжні шари. Саме це дозволяє шарам краще зберігати корисну інформацію та полегшує тренування.

Зображення, що містить текст, схема, Шрифт, знімок екрана

Автоматично згенерований опис

Рисунок 2.4 – залишкове з’єднання

ResNet має кілька варіацій, які відрізняються кількістю шарів: ResNet‑18, ResNet-34, ResNet-50, ResNet-101, ResNet-152. Число в назві вказує на кількість шарів у мережі, причому ResNet-18 є найменш глибокою архітектурою, а ResNet-152 – найглибшою. Приклад ResNet-50 зображено на рисунку 2.5.

Зображення, що містить текст, знімок екрана, Барвистість, графічний дизайн

Автоматично згенерований опис

Рисунок 2.5 – архітектура 50-шарової моделі ResNet

З переваг ResNet забезпечує високу точність класифікації, завдяки чому є одним із провідних рішень для задач комп’ютерного зору. Архітектура дозволяє ефективно навчати дуже глибокі моделі, завдяки чому можливе тренування нейронних мереж із сотнями шарів без втрати продуктивності. Також ResNet відмінно справляється з обробкою великих наборів даних, що робить її надійною для складних завдань. Гнучкість і модульність архітектури дає змогу легко адаптувати її для різних варіантів завдань

EfficientNet

EfficientNet – це сучасна архітектура нейронної мережі, яка була представлена компанією Google у 2019 році. Її основною метою є забезпечення максимальної продуктивності і ефективності обчислень, досягаючи високої точності при меншому споживанні ресурсів порівняно з іншими великими моделями для класифікації зображень.

EfficientNet-B0 – це базова версія мережі. Після базової моделі були розроблені масштабовані версії від EfficientNet-B1 до EfficientNet-B7. Кожна з цих моделей має більше шарів і фільтрів, а також обробляє зображення з більш високою роздільною здатністю, що дозволяє досягти кращих результатів на великих наборах даних. Різницю майбутніх поколінь EfficientNet у розмірах вхідних зображень показано у таблиці 2.1.

Таблиця 2.1 – різниця роздільної здатності EfficientNet

|  |  |
| --- | --- |
| **Ім’я моделі** | **Вхідне зображення** |
| EfficientNet-B0 | 224×224 |
| EfficientNet-B1 | 240×240 |
| EfficientNet-B2 | 260×260 |
| EfficientNet-B3 | 300×300 |
| EfficientNet-B4 | 380×380 |
| EfficientNet-B5 | 456×456 |
| EfficientNet-B6 | 528×528 |
| EfficientNet-B7 | 600×600 |

Основним будівельними блоками EfficientNet є інвертовані «вузькі» шари (MBConv), що показано на рисунку 2.6. Вони оптимізовані для зменшення обчислювальних витрат і використання пам’яті. Ці блоки дозволяють мережі зберігати важливі особливості зображень, водночас зменшуючи кількість обчислень.

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Рисунок 2.6 – архітектура EfficientNet-B0 з використанням MBConv

EfficientNet демонструє значну перевагу в порівнянні з іншими великими архітектурами. Наприклад, EfficientNet-B7 досягає точності понад 84% на ImageNet, при цьому використовуючи на порядок менше параметрів і обчислень. Це робить модель дуже привабливою для додатків із обмеженими ресурсами, таких як мобільні пристрої та системи вбудованого типу.

MobileNet

MobileNet розроблена компанією Google для ефективної класифікації зображень і інших задач комп’ютерного зору для використання в мобільних і вбудованих системах з обмеженими ресурсами. Вона застосовує розподілені згорткові операції, що суттєво знижує кількість параметрів і обчислювальних операцій, зберігаючи прийнятний рівень точності.

Оскільки MobileNet використовує згортки, розділені на два етапи – глибинну згортку (обробка кожного каналу окремо) і точкову згортку (об’єднання каналів), це призводить до значного зниження обчислювальної складності. MobileNet дозволяє регулювати кількість каналів і роздільну здатність мережі, що дає можливість збалансувати точність моделі та ефективність обчислень.

Архітектура MobileNet (рис. 2.7) має на даний момент 3 версії. Перша версія оптимізована для мобільних пристроїв завдяки використанню глибинно-згорткових шарів. Друга версія додає шар інверсійного залишку і лінійні шари згортки, що допомагають зменшити втрати інформації. У третій версії застосовано пошук архітектури нейронної мережі для оптимізації параметрів, а також додано блоки з активацією Swish, що додатково підвищує ефективність роботи моделі.

Зображення, що містить текст, знімок екрана, схема, Шрифт

Автоматично згенерований опис

Рисунок 2.7 – приклад архітектури MobileNetV2

Завдяки своїй ефективності, MobileNet добре підходить для реального часу та використання на пристроях з обмеженими ресурсами, де важливо мати високу продуктивність і невеликий розмір моделі.

DenseNet

Ідею щільних з’єднань між шарами пропонує DenseNet, де кожен шар отримує вхід з усіх попередніх шарів і передає свої виходи всім наступним. Це означає, що всі шари тісно з’єднані, і кожен шар отримує інформацію від усіх попередніх шарів, що сприяє зменшенню проблеми зникнення градієнта, полегшує навчання та підвищує ефективність використання параметрів мережі.

Архітектура DenseNet складається з перехідних шарів і щільних блоків. Кожен згортковий шар усередині щільного блоку пов’язаний з кожним іншим шаром у цьому блоці. Це досягається шляхом підключення виходу кожного шару до входу наступного шару, що створює «коротке» з’єднання. Перехідні шари зменшують розмір карт ознак між щільними блоками, що дозволяє мережі ефективно розширюватися. Як і в інших нейромереж, у DenseNet існує кілька версій, які різняться глибиною та кількістю параметрів.

Зображення, що містить ескіз, малюнок, схема, будинок

Автоматично згенерований опис

Рисунок 2.8 – приклад архітектури нейромережі DenseNet

Нейромережа DenseNet ефективно бореться з перенавчанням, зменшуючи кількість параметрів і забезпечуючи повторне використання ознак, що підвищує здатність моделі до узагальнення нових даних. Завдяки повторному використанню інформації, мережа може досягти високої продуктивності з меншою кількістю параметрів, ніж інші архітектури. Це дозволяє DenseNet бути економічно ефективною та швидкою при використанні обмежених обчислювальних ресурсів.

Vision Transformers

ViT є відносно новою архітектурою (рис. 2.9) для задач комп’ютерного зору, яка використовує механізми трансформерів замість згорток. ViT розбиває зображення на патчі, перетворюючи їх на послідовність, схожу на обробку тексту в моделях обробки природної мови. Хоча ViT вимагає великих наборів даних для тренування, вона демонструє високу продуктивність, перевершуючи традиційні CNN на деяких задачах.

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

Рисунок 2.9 – архітектура Vision Transformers

Механізми трансформерів (рис. 2.10) замість роботи з окремими пікселями, ViT розбиває зображення на невеликі патчі. Кожен патч обробляється як окремий елемент, подібно до токенів у тексті, після чого перетворюється у вектор ознак. Це дозволяє моделі працювати з послідовністю патчів, що є більш зручним для трансформерів.

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

Рисунок 2.10 – схема механізми трансформера

Нейромережа здатна ефективно масштабуватися з ростом кількості шарів і параметрів, що дозволяє створювати надглибокі моделі для складних задач, в тому числі і для класифікації зображень. ViT не потребує попередньо визначеної структури згорток, що робить його гнучкішим при використанні різноманітних типів вхідних даних.

Метрики вимірювання точності моделі

Точність – один із основних показників якості роботи моделі, проте в різних задачах можуть використовуватись й інші метрики. Оцінка точності – це процес вимірювання, наскільки добре модель виконує свою задачу на наборі тестових даних.

Для початку необхідно розподілити дані на тренувальну та тестову вибірки. Тренувальна вибірка використовується для навчання моделі, допомагаючи їй знайти оптимальні ваги і параметри. Тестова вибірка, в свою чергу, це набір даних, який модель ніколи не бачила під час навчання. Використовується для оцінки здатності моделі узагальнювати результати на нові дані.

Як було зазначено раніше, точність – це найпростіша і найпоширеніша метрика для задач класифікації. Вона визначає відсоток правильно передбачених класів з усіх передбачень.

Простим прикладом для пояснення розрахунку точності моделі є класифікація даних на два класи: «Positive» та «Negative». Для оцінки її результатів використовується матриця плутанини (табл. 2.2), яка відображає чотири можливі варіанти передбачень: True Positive (TP) – випадки правильної класифікації позитивного класу; True Negative (TN) – випадки правильної класифікації негативного класу; False Positive (FP) – негативний клас помилково класифікується як позитивний; False Negative (FN) – коли позитивний клас класифікується як негативний.

Таблиця 2.2 – Загальний вид матриці плутанини

|  |  |  |
| --- | --- | --- |
|  | **Прогноз: Positive** | **Прогноз: Negative** |
| Фактичне: Positive | True Positive (TP) | False Negative (FN) |
| Фактичне: Negative | False Positive (FP) | True Negative (TN) |

Точність обчислюється за наступною формулою:

де – точність.

Припустимо, що модель класифікує 100 зразків. Із них: 50 прикладів –позитивні зразки і ще 50 прикладів це негативні зразки.

Модель правильно класифікувала 40 позитивних як позитивні (TP) і 45 негативних як негативні (TN). Модель помилилася у 5 негативних випадках коли зразки класифіковано як позитивні (FP) і у 10 позитивних класифікованих як негативні (FN). Точність в даному випадку буде розраховуватися наступним чином:

Точність позитивних передбачень знаходять як частку правильних позитивних передбачень серед усіх передбачених позитивних випадків

де – точність позитивних передбачень.

Чутливість знаходять як частку правильно передбачених позитивних випадків серед усіх справжніх позитивних випадків.

де – чутливість.

F1-Score використовується в задачах класифікації, коли є необхідність збалансувати точність позитивних передбачень і чутливість, особливо у випадках, коли дані мають дисбаланс класів або коли помилкові позитивні та негативні передбачення мають різні наслідки. F1-Score є гармонійним середнім між точністю та повнотою і показує загальну продуктивність моделі і виглядає наступним чином:

де  – гармонічне середнє між Precision і Recall.

F1-Score також може використовуватись для оцінки моделі, коли потрібно запобігти перенавчанню. Якщо модель має високу точність на тренувальних даних, але низьку повноту на тестових даних, це свідчить про перенавчання. У таких випадках F1-Score допомагає оцінити загальну ефективність моделі.