Лабораторна робота №4

Згорткові нейроні мережі

Мета: ознайомитись з основними можливостями створення згорткових нейронних мереж за допомогою tensorflow

4.1 Теоретичні відомості

Згорткова нейронна мережа — це спеціалізований тип штучної нейронної мережі, яка особливо ефективна для обробки та аналізу візуальних даних, таких як зображення та відео. Згорткові нейронні мережі (ЗНМ) мають вирішальне значення в машинному навчанні , особливо в підмножині машинного навчання — глибокому навчанні .

Конвертерні мережі складаються з вузлових шарів, включаючи вхідний шар, один або кілька прихованих шарів та вихідний шар. Окремі вузли взаємопов'язані, і кожен з них має вагу та поріг, пов'язані з ним. Як тільки вихідний сигнал одного вузла перевищує заданий поріг, він активується та надсилає дані на наступний шар мережі.

Різні типи нейронних мереж використовуються для різних типів даних та випадків використання. Наприклад, рекурентні нейронні мережі часто використовуються для обробки природної мови та розпізнавання мовлення, тоді як згорткові нейронні мережі (ЗНМ) частіше використовуються для класифікації та завдань комп'ютерного зору . Здатність нейронних мереж розпізнавати складні закономірності в даних робить їх важливим інструментом у штучному інтелекті .

CNN відрізняються від інших нейронних мереж своєю чудовою продуктивністю в обробці зображень, мовлення та аудіосигналів. Вони мають три основні типи шарів, і з кожним шаром CNN стає складнішою та здатною ідентифікувати, наприклад, більші частини зображення.

Комп'ютери розпізнають зображення як числові комбінації, або, точніше, як значення пікселів . Алгоритм CNN також робить це. Наприклад, чорно-біле зображення довжиною m та шириною n представляється як двовимірний масив розміром mXn. Для кольорового зображення того ж розміру використовується тривимірний масив. Кожна комірка масиву містить відповідне значення пікселя, і кожне зображення представлено відповідними значеннями пікселів у трьох різних каналах, що відповідають червоному, синьому та зеленому каналам.

Далі визначаються найважливіші ознаки зображення. Вони витягуються за допомогою методу, відомого як згортка. Це операція, під час якої одна функція змінює (згортає) форму іншої функції. Підвищення чіткості, згладжування та покращення зображення – це поширені способи використання згорток для зображень. Однак у випадку звивистих нейронних мереж (CNN) згортки використовуються для вилучення важливих ознак із зображень.

Фільтр або ядро використовується для вилучення ключових ознак із зображення. Фільтр – це масив, який представляє певну ознаку, яку потрібно вилучити. Фільтр застосовується до вхідного масиву, а результуючий масив – це двовимірний масив, який показує, де і наскільки сильно ознака проявляється на зображенні. Вихідна матриця відома як карта ознак.

Під час процесу згортки вхідне поле перетворюється на менше поле, яке зберігає просторову кореляцію між пікселями шляхом застосування фільтрів. Нижче ми розглянемо три основні типи шарів згортки.

Згортковий шар: Цей шар є першим шаром згорткової мережі. Використовуючи фільтри (матриці з малою вагою), що ковзають по зображеннях, шар здатний розпізнавати локальні ознаки, такі як ребра, кути та текстури. Кожен фільтр створює карту ознак, яка виділяє певні візерунки. Можна використовувати більше одного згорткового шару, створюючи ієрархічну структуру в CNN, завдяки якій наступні шари можуть бачити пікселі, розташовані в рецептивних полях попередніх шарів.

Шар об'єднання: Цей шар зменшує розмір карт ознак, підсумовуючи локальні області та відкидаючи нерелевантну інформацію. Це зменшує обчислювальну складність, водночас гарантуючи збереження найважливішої інформації.

Повністю зв'язаний шар: Подібно до структури в природній нейронній мережі, цей шар з'єднує всі нейрони. Використовуваний для остаточної класифікації, він поєднує витягнуті ознаки для ідентифікації об'єкта на зображенні.

Уявімо, що ви намагаєтеся визначити, чи містить зображення людське обличчя. Можна уявити обличчя як суму його частин: два ока, ніс, рот, два вуха тощо. Ось як виглядає процес згортки.

Перші згорткові шари: Перші згорткові шари використовують фільтри для розпізнавання ознак з окремих пікселів. Наприклад, фільтр може розпізнавати вертикальний край, що представляє край ока. Як згадувалося, локальні ознаки формують візерунки, зареєстровані як карти ознак під час згортки. У цьому випадку карта ознак може представляти краї очей, носа та рота.

Додаткові згорткові шари: Після перших згорткових шарів можна застосовувати додаткові або об'єднувати шари. Наступні згорткові шари поєднують прості ознаки у складніші візерунки. Окремі візерунки поступово формують обличчя. Наприклад, краї та кути можна поєднувати у форми, подібні до очей. Додані шари бачать більші області зображення (рецептивні поля) та розпізнають складні структури, відомі як ієрархії ознак у згорткових шарах. Шар, який додається пізніше, зможе розпізнати, що коли два ока та рот розташовані певним чином, вони утворюють обличчя.

Об'єднання шарів: вони зменшують розмір карт ознак та додатково абстрагують ознаки. Це зменшує обсяг даних, які потрібно обробити, зберігаючи при цьому основні ознаки.

Повністю зв'язаний шар: Останній шар CNN – це повністю зв'язаний шар. У цьому шарі CNN створюватиме зображення людського обличчя, яке завдяки згорткам буде чітко відрізнене від іншого обличчя.

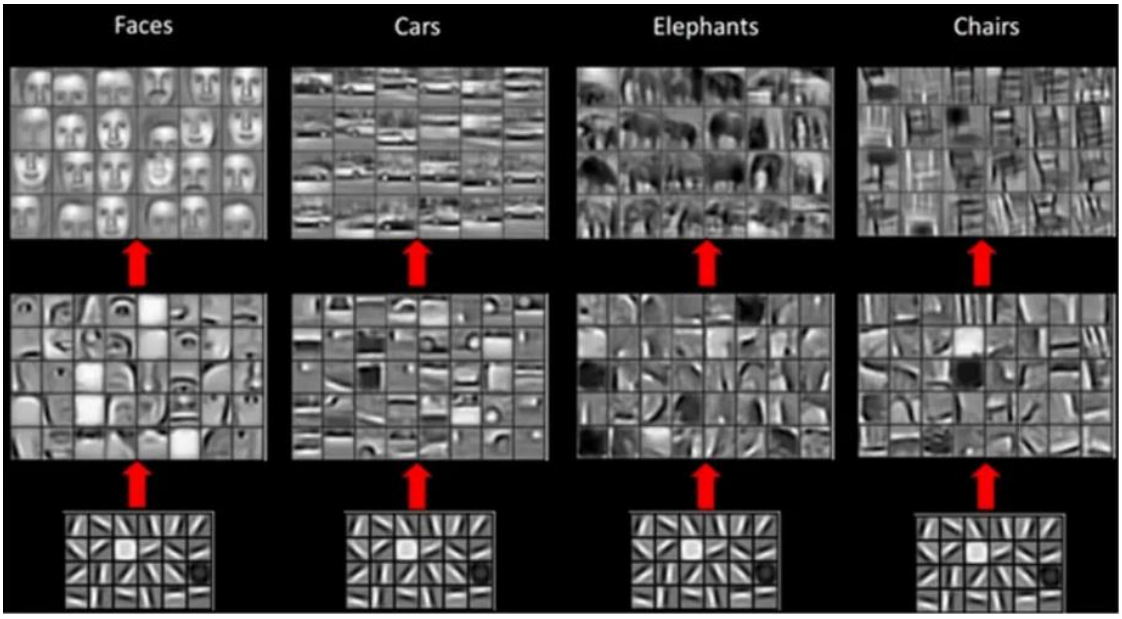

Рис.4.1 – Процес як згорткові мережі бачать різні об’єкти

Крім того, такі методи, як відсіювання та регуляризація, оптимізують CNN, запобігаючи перенавчанню. Функції активації, такі як ReLU (випрямлена лінійна одиниця), вводять нелінійність і допомагають мережі розпізнавати складніші закономірності, гарантуючи, що не всі нейрони виконуватимуть однакові обчислення. Крім того, пакетна нормалізація стабілізує та пришвидшує навчання, обробляючи дані більш рівномірно.

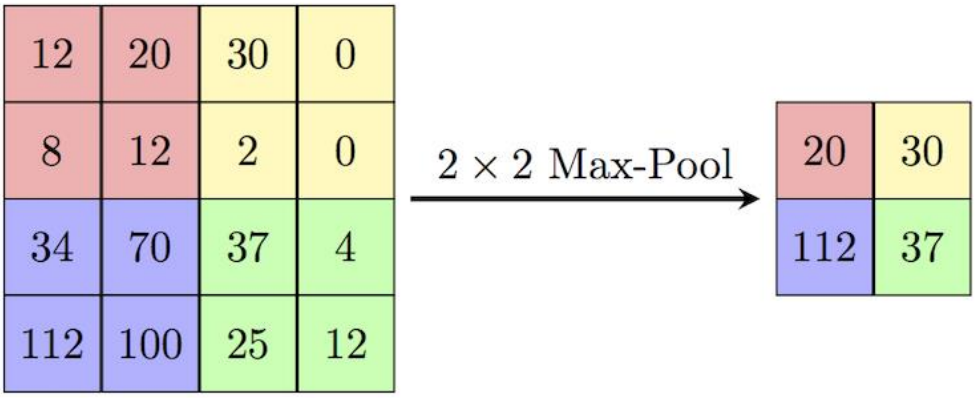

Рис.4.3 – візуалізація операції субдискретизації.

MaxPooling — це операція субдискретизації у згорткових нейромережах, яка перетворює кожне невелике вікно ознак (типово 2×2 або 3×3) на одне число — максимум у цьому вікні. У підсумку зменшується просторовий розмір карти ознак, зберігаються найсильніші відгуки фільтрів і зростає інваріантність до дрібних зсувів, шуму та локальних деформацій. Історично MaxPooling став ключем до успіху ранніх CNN (LeNet, AlexNet, VGG), бо різко знижував обчислювальну й параметричну вартість глибоких моделей без суттєвої втрати якості на задачах класифікації.

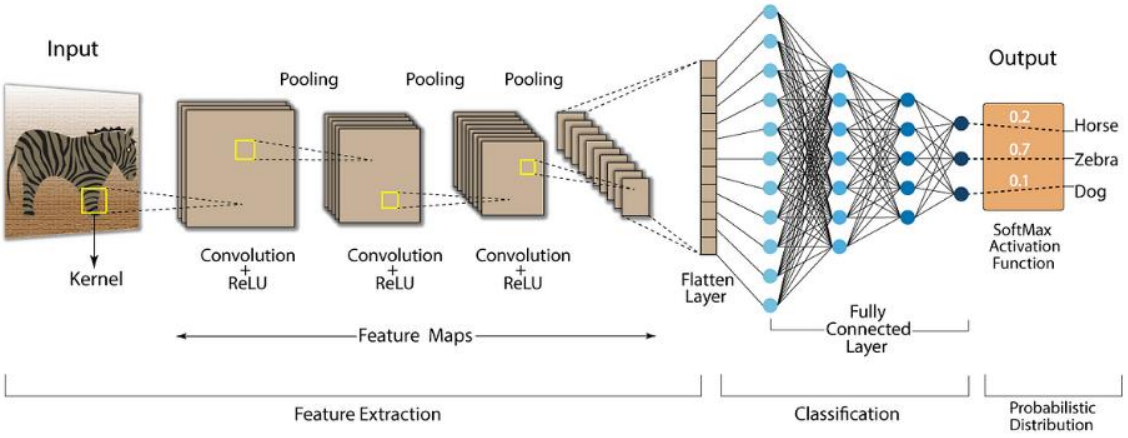

Рис.4.4 –Загальний вигляд шарів в CNN.

4.2 Порядок виконання лабораторної роботи

1. Ознайомитись з кодом в додатку 1;
2. Ознайомитись з вхідними даними датасету та вивести по одному зображенню з вибірок train/val/test;
3. Запустити процес навчання моделі та вивести графіки точності та втрат для датасету train та val;
4. Завантажити в оперативну пам’ять зображення задане викладачем та зробити передбачення чи правильно навчена модель;
5. З тестової вибірки вивести зображення (16 шт) та вивести істину мітку, визначену мітку та прогноз для зображення.

4.3 Зміст звіту

1. Найменування і мета роботи;
2.Зображення з датасетів 3шт;
3. Графіки точності та втрат;
4. Зображення задане викладачем та прогноз зображення;
5. Зображення з тестової вибірки з прогнозами мітками та істинними мітками;
6. Вивести кількість зображень у кожному спліті, середній/медіанний розмір зображень.
7. Висновки.

4.4 Контрольні питання

1. Що таке згорткова нейронна мережа (Convolutional Neural Network)?
2. У чому основна відмінність CNN від звичайних повнозв’язних мереж?
3. Що виконує операція згортки (convolution)?
4. Яку роль відіграють ядра (filters/kernels) у CNN?
5. Що таке stride і як він впливає на розмір вихідного зображення?

```
import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras import layers, models
from PIL import Image
import numpy as np


splits = ['train[:80%]', 'train[80%:90%]', 'train[90%:]']  # train/val/test
(train_ds, val_ds, test_ds), ds_info = tfds.load(
    'cats_vs_dogs',
    split=splits,
    as_supervised=True,    # (image, label)
    with_info=True
)

IMG_SIZE = 128
BATCH = 32
AUTOTUNE = tf.data.AUTOTUNE

def preprocess(img, label):
    img = tf.image.resize(img, (IMG_SIZE, IMG_SIZE))
    img = tf.cast(img, tf.float32) / 255.0
    return img, label

train_ds = (train_ds
            .shuffle(10_000)
            .map(preprocess, num_parallel_calls=AUTOTUNE)
            .batch(BATCH)
            .prefetch(AUTOTUNE))

val_ds = (val_ds
          .map(preprocess, num_parallel_calls=AUTOTUNE)
          .batch(BATCH)
          .prefetch(AUTOTUNE))

test_ds = (test_ds
           .map(preprocess, num_parallel_calls=AUTOTUNE)
           .batch(BATCH)
           .prefetch(AUTOTUNE))

model = models.Sequential([
    layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3)),
    layers.Conv2D(16, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
```

```
    loss='binary_crossentropy',
    metrics=['accuracy']
)

model.summary()
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=3
)
def predict(path):
    img = Image.open(path).convert("RGB").resize((IMG_SIZE, IMG_SIZE))
    x = np.array(img, np.float32)/255.0
    p = model.predict(x[None], verbose=0)[0,0]
    print("dog" if p>=0.5 else "cat", float(p))

predict("dog.jpg")
```