ПРИВАТНИЙ ЗАКЛАД ВИЩОЇ ОСВІТИ «ІТ СТЕП Університет»

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи бакалавра

Цюра Павло Ігорович

студент групи 4CS-41, спеціальності 122 Комп’ютерні науки

**Тема:**

**Згорткові нейронні мережі для вирішення задачі опрацювання зображень за умов поганої видимості**

Изображение выглядит как лазер, ночное небо

Автоматически созданное описание

Науковий керівник: Дейнеко А.О., доцент, к.т.н.\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Консультант: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кваліфікаційна робота захищена з оцінкою «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Секретар ЕК \_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

Львів – 2022

ПРИВАТНИЙ ЗАКЛАД ВИЩОЇ ОСВІТИ «ІТ СТЕП Університет»

ЗАТВЕРДЖУЮ

Проректор з науково-педагогічної роботи

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_року

**З А В Д А Н Н Я**

**НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

Освітній рівень – бакалавр

Спеціальність - 122 Комп’ютерні науки

Цюра Павло Ігорович

(прізвище, ім’я, по батькові)

1. Тема роботи: “Розробка інтелектуальної системи виправлення граматичних помилок для української мови”

Керівник роботи: Дейнеко Анастасія Олександрівна, доцент, к.т.н

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

затверджені наказом Університету від “\_\_\_”\_\_\_\_\_\_\_\_\_\_20\_\_року №\_\_\_

2. Строк подання студентом роботи: 26.06.2022

3. Вихідні дані до роботи (назва алгоритмів, методів, технологій, назви вибірок даних тощо)

Багатошаровий пецептрон, згорткова нецронна мережа, ResNet, ResUNet, система для покращення якості зображеннь, навчання нейронної мережі, GLEU, Python, TensorFlow, пермутація, комбінація, трансферне навчання, оптимізація гіперпараметрів, тонке налаштування,

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

Розділ 1. Аналіз предметної галузі та постановка задачі дослідження; Розділ 2. Згорткові нейронні мережі для вирішення задачі покращення зображеннь Розділ 3. Імітаційне моделювання

5. Перелік слайдів презентації, схем тощо

Тема, актуальність та постановка задачі, основні завдання, теоретичні дослідження, імітаційне моделювання, висновки, апробація роботи

**ЕТАПИ ТА ТЕРМІНИ ПІДГОТОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | ПЕРЕЛІК РОБІТ | Термін виконання | Дата ознайомлення наукового керівника | Підпис наукового керівника | Примітки |
| 1. | Вибір теми, її затвердження та закріплення наукового керівника | грудень | 21.12.2021 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 2. | Складання плану роботи та узгодження з науковим керівником | січень | 30.01.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 3. | Аналіз літературних джерел, матеріалів періодичних видань, збір та узагальнення даних та обговорення з науковим керівником | лютий | 15.02.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 4. | Написання розділів кваліфікаційної роботи. Постановка експерименту, аналіз отриманих результатів наукового дослідження | Квітень-травень | 05.04.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 5. | Проміжний контроль виконання роботи | Кожен тиждень |  | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 6. | Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику | червень | 21.06.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 7. | Подання на зовнішню рецензію | червень | 21.06.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 8. | Підготовка до захисту кваліфікаційної роботи: написання доповіді та розробка презентації | червень | 20.06.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
| 9. | Подання кваліфікаційної роботи до навчально-методичного відділу з усіма супроводжувальними документами | червень | 21.06.2022 | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание | Вик. |
|  | Публічний захист кваліфікаційної роботи перед Атестаційною комісією | Згідно з розкладом  роботи ЕК |  | Изображение выглядит как лазер, ночное небо  Автоматически созданное описание |  |

Графік узгоджено «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Здобувач: Цюра Павло Ігорович

Науковий керівник: Дейнеко Анастасія Олександрівна

**календарний план**

**ЗМІСТ**

|  |  |
| --- | --- |
| Перелік умовних позначень, символів, одиниць, скорочень і термінів..... | 8 |
| Вступ………………………………………………………………………… | 9 |
| 1 Аналіз предметної галузі та постановка задачі дослідження…………... | 11 |
| 1.1 Адаліна……………………………………………………………….. | 12 |
| 1.2 Персептрон Розенблатта…………………………………………….. | 13 |
| 1.3 Багатошаровий персептрон………………………………………….. | 14 |
| 1.4 Глибинні нейронні мережі…………………………………………... | 18 |
| 1.5 Задача розпізнавання образів………………………………………... | 21 |
| 1.6 Постановка задачі дослідження……………………………………... | 22 |
| 2 Згорткові нейронні мережі для розпізнавання образів…………………. | 23 |
| 2.1 Згорткові нейронні мережі…………………………………………... | 23 |
| 2.2 Конволюційна нейронна мережа типу ResNet……………………… | 36 |
| 3 Імітаційне моделювання ………………………………………………… | 41 |
| Висновки…………………………………………………………………….. | 59 |
| Перелік джерел посилання…………………………………………………. | 60 |
| Додаток А Вихідний код програми………………………………………… | 64 |
| Додаток Б Відомість кваліфікаційної роботи……………………………... | 66 |

# Перелік умовних позначень, символів, одиниць, скорочень і термінів

ШНМ – штучна нейрона мережа;

BN – batch normalisation – пакетна нормалізація;

CNN – convolutional neural network – згорткова нейронна мережа;

CONV – convolutional layer – шар згортки;

ResNet – residual network – залишкова мережа;

MAXPOOL – max pooling layer – шар максимальної субдискретизації.

SSIM – structural simularity matrix – матриця структурної схожості

MSE – mean squared error – середньоквадратична похибка

GELU – gaussian error linear unit - Лінійна одиниця похибки Гауса

# Вступ

На сьогоднішній день технології штучних нейронних мереж (ШНМ) досить часто і досить успішно використовуються у вирішенні складних задач, які, як правило, не мають аналітичного рішення. На сьогодні нейронні мережі стають все поширенішими для вирішення різних задач обробки сигналів, оптимізації, оптимального і адаптивного управління, розпізнавання образів, ідентифікації, прогнозування в реальному часі і т.п. Створено реальні системи обробки зображень та комп'ютерного зору, управління аерокосмічними об'єктами, технічної та медичної діагностики, в економіці і фінансах (планування, управління, аналіз ринків, прогнозування курсів, технічний аналіз, пошук інформації, ідентифікація кредитних карт), у військовій справі, управлінні рухом, в енергетиці (оцінка станів, виявлення розладнань, прогнозування навантажень), в криміналістиці, аналізі сигналів різної природи та ін., причому цей перелік розширюється практично щодня. Однією з перспективних завдань є обробка зображень за допомогою згорткових нейронних мереж.

Люди давно замислювалися про те, як саме ми бачимо навколишній світ, як працює зір. Зв'язок очей із зором була, вибачте за каламбур, очевидною навіть древнім. Око як оптичний прилад вивчали Леонардо да Вінчі, Йоганн Кеплер і багато інших великі фізики, що відзначали його видатні оптичні властивості.

Завдання обробки зображень – це прийом початкового зображення і виведення його класу (кішка, собака і т.д.) або групи ймовірних класів, яка найкраще характеризує зображення. Для людей це один з перших навичок, який вони починають освоювати з народження.

Згорткові нейронні мережі (convolutional neural networks, CNN) – це досить широкий клас архітектур, основна ідея яких полягає в тому, щоб перевикористати одні і ті ж частини нейронної мережі для роботи з різними маленькими, локальними ділянками входів. Як і багато інших нейронні архітектури, згорткові мережі відомі досить давно, і в наші дні у них вже знайшлося багато найрізноманітніших застосувань, але основним додатком, заради якого люди колись придумали згорткові мережі, залишається обробка зображень. І це не випадково: ідея згортальних мереж багато в чому мотивована дослідженнями про зоровій корі головного мозку.

Сьогодні глибинне навчання лежить в основі послуг багатьох компаній: Facebook використовує нейронні мережі для алгоритмів автоматичного створення тегів, Google – для пошуку серед фотографій користувача, Amazon – для генерації рекомендацій товарів, Pinterest – для персоналізації домашньої сторінки користувача, а Instagram – для пошукової інфраструктури.

# 1 Аналіз предметної галузі та постановка задачі дослідження

Штучні нейронні мережі (ШНМ), що являють собою один із напрямів обчислювального інтелекту ‒ це потужний інструмент для вирішення широкого кола завдань інтелектуального аналізу даних, які можна звести до трьох основних типів: апроксимація, кластерування, асоціативна пам'ять. З плином часу ця методологія активно і успішно використовувалася дослідниками в різних сферах діяльності людини. На сьогоднішній день розроблено безліч типів штучних нейронів, архітектур ШНМ з різною топологією, алгоритмів навчання, що дозволяють робити налаштування синаптичних ваг за короткі проміжки часу і обробляти великі потоки даних в потоковому режимі, постійно «донавчатися» на даних, що надходять, при цьому традиційні нейронні мережі в загальному випадку є системою типу «чорний ящик» і не дозволяють інтерпретувати отримані результати. Загалом під процесом навчання нейронної мережі традиційно розуміють процедуру адаптації синаптичних вагових коефіцієнтів для вирішення поставленого завдання шляхом оптимізації прийнятого критерію якості. Треба зауважити, що метою навчання є отримання архітектури та параметрів мережі, яка буде найкращим чином відтворювати заздалегідь невідоме відображення. Цим невідомим відображенням може бути залежність вихідних параметрів будь-якого процесу від вхідних, прогнозу від передісторії, класу об'єкту від набору його властивостей, керуючого впливу від поточного стану об'єкта управління і т.д. Однак, крім синаптичних ваг, налаштовувати можна і кількість шарів в мережі або кількість штучних нейронів всередині кожного шару.

Основною особливістю штучних нейронних мереж і їх нейронів є здатність до навчання, в процесі якого синаптичні ваги налаштовуються за допомогою того чи іншого адаптивного алгоритму з метою найбільш ефективного вирішення поставленої проблеми.

## 1.1 Адаліна

Одним з найпростіших нейронів який має здатність до навчання є адаптивний лінійний елемент (ADALINE), запропонований Б. Уїдроу і наведений на рисунку 1.1.

Рис2

Рисунок 1.1 – Адалина

Структурно АДАЛІН вельми нагадує нейрон Маккаллоха-Питтса з активационной Сігнум-функцією і складається з двох основних частин: адаптивного лінійного асоціатор і нелінійної активаційної функції. АДАЛІН має входів – і два виходи – аналоговий і бінарний . Крім того, є додатковий вхід, на який подається навчальний сигнал , що показує якою має бути бажана реакція нейрона на кожен конкретний набір вхідних сигналів. Аналоговий вихід являє собою зважену суму входів як показано у формулі (1.1)

|  |  |
| --- | --- |
|  | (1.1) |

а бінарний вихід може приймати значення  або  в залежності від полярності аналогового сигналу . Вихідний сигнал порівнюється із зовнішнім навчальним сигналом і виникає при цьому сигнал помилки надходить в алгоритм навчання, який перебудовує синаптичні ваги так, щоб мінімізувати деяку функцію помилки , звану критерієм навчання. В якості такої функції найчастіше використовують квадратичну, що дозволяє використовувати для навчання не тільки «рідний» алгоритм, синтезований Б. Уїдроу і М. Хоффом спеціально для АДАЛІН, а й безліч зворотних процедур адаптивної ідентифікації.

АДАЛІН може використовуватися як в якості елементарного нейрона в складі ШНС, так і самостійно в задачах розпізнавання образів, обробки сигналів, реалізації логічних функцій.

## 1.2 Персептрон Розенблатта

Елементарний персептрон Ф. Розенблатта структурно подібний до АДАЛІН, що видно з його схеми, наведеної на рисунку 1.2.

Основна відмінність полягає в алгоритмі навчання, оскільки в АДАЛІН помилка є лінійною функцією від векторів входів , в той час як помилка навчання персептрона від входів залежить нелінійно, причому характер цієї нелінійності визначається активаційною функцією , яка в загальному випадку може бути недиференційованої (наприклад, Сігнум-функція). Дана обставина істотно ускладнює процес навчання персептрона і вимагає використання спеціальних процедур.

Рис2

Рисунок 1.2 – Персептрон Розенблатта

Сплеск досліджень, пов'язаних з елементарними персептрони припав на 60-і роки. Надалі ці роботи практично припинилися після публікації монографії М. Мінського і С. Пейперта, які показали досить обмежені можливості елементарного персептрона при вирішенні задач класифікації або реалізації логічних функцій (проблема «виключає АБО»).

## 1.3 Багатошаровий персептрон

Новий сплеск інтересу до персептронів доводиться на кінець 80-х років. Саме до цього періоду відноситься поява багаторівневих штучних нейронних мереж і алгоритмів їх навчання. Як уже зазначалося, властивості ШНС значною мірою визначаються їх топологією (архітектурою). На сьогодні більшість існуючих нейронних мереж в залежності від архітектури може бути розбите на три великі категорії: багатошарові з прямою передачею інформації, в яких окремі нейрони об'єднані в шари, між якими інформація передається в одному напрямку від входу до виходу (рисунок 1.3 а), рекурентні (зі зворотним зв'язком), в яких сигнали з виходу можуть подаватися на вхід або внутрішні шари мережі (рисунок 1.3 б) і клітинні, в яких кожен нейрон пов'язаний тільки зі своїми сусідами (рисунок 1.3 в).

Найбільш яскравим представником ШНС з прямою передачею інформації є багатошарові персептрони. У загальному випадку кожен нейрон даного рівня однонаправленно пов'язаний з усіма нейронами наступного рівня. Ці зв'язки організовані через синаптичні ваги, які діють як підсилювачі в відповідних каналах. Наприклад, в тришаровому персептрони, схема якого наведена на рисунку 1.4 а (сигнали зміщення для простоти зображення опущені), нейрони згруповані в послідовні шари: нульовий, перший, другий і третій. Нейрони нульового шару, іноді званого вхідним, не виробляють ніяких обчислень, а служать лише для передачі вектора вхідних сигналів на нейрони шару, званого першим прихованим шаром. Сигнали з першого шару надходять на другий прихований шар і далі на третій. Останній третій шар є вихідним і в ньому формується вектор вихідних сигналів ШНС в цілому. Теоретично кількість шарів може бути довільно великим, проте поруч авторів було доведено, що для вирішення як завгодно складних завдань апроксимації і розпізнавання образів досить лише трьох шарів, тобто тришаровий персептрон є універсальним аппроксіматором.

Рис2

Рис2

Рис2

Рисунок1.3 – Архітектури нейронних мереж

Зауважимо, що в багатошарових персептрона зв'язку між нейронами всередині шару або від вищого до нижчого прошарку заборонені, при цьому кількість нейронів, а відповідно і синаптичних ваг в кожному шарі по-різному. Кожен нейрон (а це, як правило, елементарний персептрон Розенблатта) має один вихід і безліч входів, які в свою чергу є виходами всіх нейронів попереднього шару.

Позначивши внутрішній сигнал -го нейрона -го шару, що містить нейронів, як , нескладно уявити його у вигляді зваженої суми виходів попереднього шару як показано у виразі (1.2)

|  |  |
| --- | --- |
|  | (1.2) |

де синаптичні ваги, що зв'язують кожен нейрон -го шару з усіма нейронами попереднього шару так, що .

Тоді вихід кожного нейрона є нелінійне перетворення, виду формули (1.3)

|  |  |
| --- | --- |
|  | (1.3) |

де для скорочення запису величена зміщення позначається як синаптична вага а .

Використовуючи векторно-матричну форму, можна записати перетворення, здійснюване кожним шаром, в компактному вигляді

|  |  |
| --- | --- |
| , | (1.4) |

де вихідний вектор сигналів;

вхідний вектор;

матриця синаптичних ваг;

діагональний нелінійний оператор, утворений активаційними функціями всіх нейронів .

З урахуванням введених позначень рівняння тришарового персептрона набирає вигляду формули (1.5)

|  |  |
| --- | --- |
|  | (1.5) |

а сама ШНС може бути представлена в компактній формі, наведеної на рисунку 1.4 б.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.4 – Багатошаровий персептрон:

а) розгорнута форма архітектури; б) комапктний вигляд

На сьогодні багатошаровий персептрон, мабуть, найбільш поширена нейросеть, що знайшла застосування в багатьох областях. Однак для того, щоб забезпечити необхідну поведінку ШНС в процесі перетворення вхідних сигналів у вихідні, необхідно відповідним чином організувати процедуру навчання.

## 1.4 Глибинні нейронні мережі

До теперішнього моменту ми розглянули вже дуже багато про нейронних мережах. Ми визначили, що вони складаються з нейронів, розібрали, як виглядає один нейрон, як вони з'єднуються в єдину нейронну мережу і, головне, як потім можна навчати цю нейронну мережу: для цього використовуються різні варіанти градієнтного спуску, а щоб власне підрахувати градієнт, можна скористатися алгоритмами автоматичного диференціювання на графі обчислень. Всі ці методи, взагалі кажучи, ніяк не залежать від архітектури нейронної мережі і теоретично повинні працювати для будь-якого графа обчислень, аби ми вміли прокидати градієнти через вузли, тобто реалізовувати процедуру зворотного поширення.

Ідея графа обчислень не здається такою вже складною, про те, як диференціювати композицію функцій, люди знають уже протягом кількох століть, та й безпосередньо алгоритм градієнтного спуску був відомий навіть раніше XX століття. Сама ідея побудувати глибоку мережу, тобто нанизати один на одного кілька рівнів нейронів, теж не здається верхом винахідливості.

Звичайно, ні. Ідеї глибоких нейронних мереж мають майже таку ж довгу історію, як і самі штучні нейронні мережі1. Перші глибокі мережі з'явилися ще в середині 1960-х років.

Перші справжні глибокі мережі у вигляді глибоких перцептронів були засновані на методі групового урахування аргументів, суть якого виглядає приблизно так:

– спочатку ми вибираємо загальний вигляд, параметричне сімейство моделей, які будемо навчати; було запропоновано використовувати так звані поліноми Колмогорова – Габора, тобто по суті просто многочлени з невідомими коефіцієнтами, але можуть бути і будь-які інші;

– будуємо і навчаємо різні варіанти обраних моделей;

– вибираємо за допомогою метрики якості кілька кращих моделей; якщо потрібну якість вже досягнуто, можна нічого далі не робити;

– але якщо ще не досягнуто – і це ключовий момент – то ми починаємо будувати моделі наступного рівня, використовуючи виходи підібраних на попередньому кроці моделей як входи для наступних;

– цей процес можна рекурсивно повторювати до тих пір, поки якість моделі або не досягне потрібного рівня, або не перестане поліпшуватися.

Метод групового урахування аргументів виглядає напрочуд сучасно. Якщо в ньому в якості базової моделі вибрати перцептрон, ми отримаємо типову нейронну мережу з декількома шарами, яка навчається шар за шаром: спочатку перший, потім він фіксується і починається навчання другого, і т.п. Вже на початку 1970-х років цим методом цілком успішно навчалися моделі аж до семи рівнів в глибину, і це дуже схоже на процедуру предобученія без вчителя.

Але все-таки нейронні мережі в результаті пішли трошки іншим шляхом. Першою глибокої нейронною мережею можна вважати вже згадуваний Neocognitron Куніхіро Фукусіми, в якому з'явилися і згорткові мережі, і активації, дуже схожі на ReLU.

Однак ця модель не навчалася в сучасному розумінні цього слова: ваги мережі встановлювалися з локальних правил навчання без учителя. Приблизно в той же час з'явилися і глибокі моделі на основі зворотного поширення. Першим застосуванням зворотного поширення помилки до довільних архитектурам можна вважати роботи фінського тоді ще студента Сеппо Лінненмаа: в 1970 році він побудував правила автоматичного диференціювання по графу обчислень.

А зараз перейдемо до питання, так би мовити, телеологического характеру: навіщо взагалі потрібні глибокі мережі? Класична теорема Хорніка, заснована на більш ранніх роботах Колмогорова, стверджує, що будь-яку безперервну функцію можна як завгодно точно наблизити нейронною мережею з одним прихованим рівнем.

А інша сторона сили глибоких мереж – це те, що глибока нейронна мережа створює не просто глибоке, але ще і розподілене представлення. Тут мова йде про те, що кожен рівень глибокої мережі складається не з одного нейрона, а відразу з багатьох, і комбінації значень цих нейронів виробляють справжнісінький експонентний вибух в просторі входів.

## 1.5 Задача розпізнавання образів

Завдання розпізнавання образів – це завдання віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних.

Розпізнавання образів є однією з найбільш фундаментальних проблем теорії інтелектуальних систем. Крім того, завдання розпізнавання образів має величезне практичне значення. Замість терміна "розпізнавання" часто використовується інший термін – "класифікація". Ці два терміни у багатьох випадках розглядаються як синоніми, але не є повністю взаємозамінними. Кожен з цих термінів має свої сфери застосування, і інтерпретація обох термінів часто залежить від специфіки конкретної задачі.

Розпізнавання образів – це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних.

При постановці завдань розпізнавання намагаються користуватися математичною мовою, намагаючись, на відміну від теорії штучних нейронних мереж, де основою є отримання результату шляхом експерименту, замінити експеримент логічними міркуваннями і математичними доказами.

Найчастіше в задачах розпізнавання образів розглядаються монохромні зображення, що дозволяє розглядати зображення як функцію на площині. Якщо розглянути безліч точок на площині T, де функція x (x, y) висловлює в кожній точці зображення його характеристику – яскравість, прозорість, оптичну щільність, то така функція є формальним записом зображення.

І багато всіх можливих функцій x(x, y) на площині T – є моделлю безлічі всіх зображень X. Вводячи поняття подібності між образами можна поставити завдання розпізнавання. Конкретний вид такої постановки сильно залежить від наступних етапів при розпізнаванні згідно того чи іншого підходу.

## 

## 1.6 Покращення якості зображень на основі глибинних нейронних мереж

## Одним з напрямків розвитку машинного навчання що почав стрімко розвиватись завдяки глибинним нейронним мережам є покращення якості зображень. Можна виділити задачі збільшення роздільної здатності та усунення шумів.

## Супер-резолюція передбачає збільшення роздільної здатності зображення. Такі моделі знаходять своє використання у обробці медичних зображень, системах спостереження та астрономічних дослідженнях. Існують підходи вирішення цієї задачі без використання машинного навчання, такі як білінійна інтерполяція та інтерполяція методом найближчого сусіда. Останній передбачає просте копіювання пікселів. Білінійна інтерполяція використовує зважену суму пікселів. Такі методи знаходять своє використання, але дуже обмежене, наприклад для збільшення штрих-кодів. Використання машинного навчання зумовлюється необхідністю фотореалістичності такого перетворення. Такі нейронні мережі мають архітектуру у якій з кожним наступним шаром збільшується розмірність даних.

#### Зниження шумів різної природи, зокрема опадів, стрімко розвивається. Як і у випадку з супер-резолюцією, існують методи які не передбачають машинного навчання, такі як розмивання Гауса, Welvet Denoising. Такі методи мають широке поле використання, але поступаються нейронним мережам через свою лінійність. Цей напрямок глибинного навчання перебуває на етапі стрімкого розвитку, оскільки може бути використаний для покращення роботоздатності безлічі інтелектуальних систем, зокрема автоматичного керування автомобілем та виявлення неналежного використання засобів особистої безпеки на будівницві.

## 1.7 Постановка задачі дослідження

Проведені попередні дослідження під час аналізу предметної області показали, що станом на 2021-й рік, розпозновання зображеннь з поганою якістю перебуває у стані активного розвитку. Як результат, ці технології активно використовуються у різних сверах. Таким чином, на сьогоднішній день актуальною є задача розробки та реалізації згорткової нейронної мережі для зображень за умов поганої видимості.

Метою даної роботи є розробка моделі згорткової глибинної нейронної мережі с використанням мережі типу ResUNet для опрацювання зображень, які були відзняті за умов поганої видимості.

Досягнення поставленої мети здійснюється шляхом вирішення наступних основних завдань:

– аналіз систем для покращення якості зображень;

– збір даних та їх попередній аналіз;

– навчання згорткової глибинної нейромережі типу ResUNet;

– аналіз отриманних результатів та «тонке» налаштування моделі нейромережі.

Об'єкт дослідження – процес покращення якості входового зображення згортковою нейронною мережею типу ResUNet.

Предмет дослідження – методи розпізнавання образів та класифікації глибинними нейронними мережами для опрацювання зображень, які були зроблені за умов погадої видимості.

Методи дослідження: теорія штучних нейронних мереж; теорія глибинних нейронних мереж; імітаційне моделювання.

# 2 Згорткові нейронні мережі ДЛЯ вирішення задачі покращення зображеннь

## 2.1 Згорткові нейронні мережі

Здатність навчатися є основною властивістю біологічного мозку, а штучна нейронна мережа в деякому сенсі моделює мозок, поняття «навчання» посідає щонайперше місце в теорії штучних нейронних мереж. Математичні проблеми, що пов’язані з навчанням, вивчають у напрямі загальної теорії штучних нейронних мереж, який дістав назву «нейроматематика». Із точки зору нейроматематики, навчання тлумачать як завдання адаптувати параметри, а можливо, й архітектуру мережі, щоби, оптимізуючи прийнятий критерій якості, розв’язати поставлену задачу. Таке визначення є узвичаєним та неявно припускає, що нейроматематика ґрунтується на методах оптимізації та ідентифікації.

Зазвичай припускають, що навчання має перманентний характер та з часом мережа покращує свої характеристики, постійно «наближаючись» до оптимального розв’язку поставленої задачі. Тип та характер навчання обумовлені, насамперед, обсягом попередньої та поточної інформації про довкілля, в яке «занурили» мережу, а також критерієм якості (цільовою функцією), що характеризує рівень відповідності нейронної мережі до розв’язуваної нею задачі. Інформацію про довкілля здебільшого задають у вигляді навчальної вибірки образів або зразків, оброблюючи їх мережа дістає відомості, необхідні для отримання шуканого розв’язку. Саме характер та обсяг цієї інформації визначають тип і методнавчання.

З погляду математики, навчання нейронних мереж – це багато-параметрична задача нелінійної оптимізації. Більшість методів навчання можна розділити на два класи: навчання з учителем (із заохоченням) та навчання без учителя (без заохочення, або самонавчання). Методи навчання з учителем застосовують у випадках, коли відома бажана реакція системи в кожну мить часу, себто відомий навчальний сигнал, який впливає на налаштування параметрів системи, що навчається. Рівень «навченості» системи формально визначають за значенням цільової функції, тобто за тим станом, якого має набути системав результаті навчання.

Перед тим, як розбирати, що таке згортка і яке вона має відношення до нейронних мереж, відступимо на крок назад. Наріжним каменем усіх нейронних мереж є аффінниє перетворення. У кожному шарі повно-мережі повторюється одна і та ж операція: на вхід подається вектор, який множиться на матрицю ваг, а до результату додається вектор вільних членів; тільки після цього до результату застосовується якась нелінійна функція активації. І у всіх мережах такий підхід використовувався постійно, незалежно від структури або походження даних. Будь то зображення, текст або музика, ми знову і знову застосовуємо Афінний перетворення в кожному шарі нашої мережі, попередньо навівши дані до векторів.

Однак багато типів даних мають свою власну внутрішню структуру, яка добре відома нам заздалегідь. У нашому випадку приклад такої структури – зображення, яке зазвичай представляють як масив векторів чисел: якщо зображення чорно-біле, то це просто масив інтенсивностей, а якщо кольорове, то масив векторів з трьох чисел, що позначають інтенсивності трьох основних кольорів (червоного, зеленого і чорного в стандартному RGB, синього, зеленого і червоного в трьох типах колбочок в людському оці і т. д.). Якщо ж узагальнити таку внутрішню структуру до максимальної все ще корисною нам спільності, опис вийде таке:

1) вихідні дані представляють собою багатовимірний масив («тензор»);

2) серед розмірностей цього масиву є одна або більше осей, порядок уздовж яких грає важливу роль; наприклад, це може бути розташування пікселів в зображенні, тимчасова шкала для музичного твору, порядок слів або символів у тексті;

3) інші осі позначають «канали», що описують властивості кожного елемента за попереднім подмножеству осей; наприклад, три компонента для зображень, два компонента (правий і лівий) для стереозвуку і т. п.

Коли ми навчаємо повнозв'язні нейронні мережі, це додаткове знання про структуру завдання ніяк не використовується. Якщо ми не будемо робити вигляд, що нічого не знаємо про структуру входів (поділі колірних каналів зображення, порядку нуклеотидів в ДНК і т.д.), а станемо її безпосередньо використовувати, це може істотно допомогти нам у навчанні нейронних мереж. У даній роботі ми розглянемо такий підхід на прикладі згортальних шарів для задач комп'ютерного зору. Ми обговоримо дуже важливу частину сучасних глибоких нейронних мереж і знову побачимо, як дуже проста концепція призводить до фантастичних результатів.

Основна ідея зготкової мережі полягає в тому, що обробка ділянки зображення дуже часто має відбуватися незалежно від конкретного розташування цієї ділянки. Грубо кажучи, якщо ви хочете дізнатися на фотографії свого друга Васю, абсолютно не важливо, на 100 або на 200 пікселів вухо Васі відстоїть від лівого краю фотографії. Дізнатися Васю можна було б і на сильно обрізаної фотографії, де немає нічого, крім його особи; це локальна задача, яку можна вирішувати локальними засобами. Звичайно, взаємне розташування об'єктів грає важливу роль, але спочатку їх потрібно в будь-якому випадку розпізнати, і це розпізнавання – локально і незалежно від конкретного положення ділянки з об'єктом всередині великий картинки. Тому згорткова мережа просто робить це припущення в явному вигляді: давайте покриємо вхід невеликими вікнами (скажімо, 5 × 5 пікселів) і будемо виділяти ознаки в кожному такому вікні невеликий нейронною мережею. Причому – і тут ключове міркування – ознаки будемо виділяти в кожному вікні одні й ті ж, то є маленька нейронна мережа буде всього одна, входів у неї буде всього 5 × 5 = 25, а з кожної картинки для неї може вийти дуже багато різних входів .

Потім результати цієї нейронної мережі знову можна буде подати у вигляді «картинки», замінюючи вікна 5 × 5 на їх центральні пікселі, і на ній можна буде застосувати другий згортковий шар, з уже інший маленької нейронною мережею, і т.д. Далі буде розглянуто, що в кожному згортковому шарі буде зовсім небагато вільних параметрів, особливо в порівнянні з повнозв'язним аналогами. Перш ніж переходити безпосередньо до формальних визначень операції згортки, давайте розберемося з поняттям каналу в зображенні. Зазвичай кольорові картинки, що подаються на вхід нейронної мережі, представлені у вигляді декількох прямокутних матриць, кожна з яких задає рівень одного з колірних каналів в кожному пікселі зображення. Картинка розміром 200 × 200 пікселів – це насправді 120 000 чисел, три матриці інтенсивностей розміром 200 × 200 кожна. Якщо зображення чорно-біле, як, наприклад, в MNIST, то така матриця буде одна. А якщо це не проста картинка, а, скажімо, результат зображує мас-спектрометрії, коли в кожному пікселі знаходиться цілий спектр, то матриць може бути дуже багато. Але в будь-якому випадку ми будемо припускати, що в кожному пікселі вхідного зображення варто деякий тензор (зазвичай одновимірний, тобто вектор чисел), і його компоненти називаються каналами (channels). Такі ж матриці будуть виходити і після згорткового шару: в них як і раніше буде просторова структура, відповідна вихідній картинці (але не в точності така ж – скоро про це поговоримо), проте каналів тепер може стати більше. Значення кожної ознаки, які ми виділили з вікон в оригінальному документі, тепер будуть являти собою цілу матрицю. Кожна така матриця називається картою ознак (feature map). В принципі, канали вихідного зображення можна теж називати картами ознак; аналогічно ми часто будемо називати карти ознак чергового шару каналами.

Тепер залишилося тільки формально визначити, що ж таке згортка і як влаштовані шари згорткової мережі. Згортка – це всього лише лінійне перетворення вхідних даних особливого виду. Якщо xl – карта ознак в шарі під номером l, то результат двовимірної згортки з ядром розміру (2d+1) і матрицею ваг W розміру (2d+1)×(2d+1) на наступному шарі буде як зазаначено виразом (2.1):

|  |  |
| --- | --- |
|  | (2.1) |

де yli,j – результат згортки на рівні l;

xli,j – її вхід, тобто вихід за все попереднього шару.

Інакше кажучи, щоб отримати компоненту (i, j) наступного рівня, ми застосовуємо лінійне перетворення до квадратному вікна попереднього рівня, тобто скалярно множимо пікселі з вікна на вектор згортки. Це проілюстровано на рисунку 2.1: ми застосовуємо згортку з матрицею ваг W розміру 3×3 до матриці X розміру 5×5. Зверніть увагу, що множення подматріци вихідної матриці X, відповідної вікна, і матриці ваг W – це не множення матриць, а просто скалярний добуток відповідних векторів. А всього вікно уміщається в матриці X дев'ять разів, і виходить

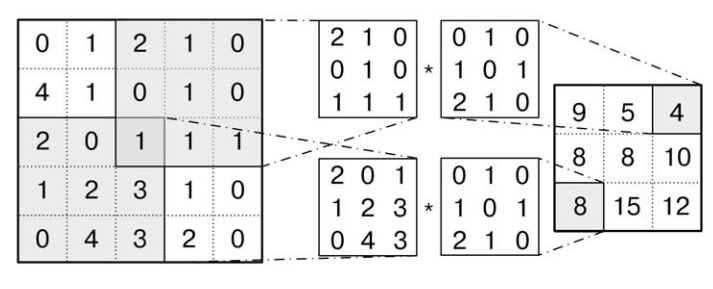


Рисунок 2.1 – Приклад підрахунку результату згортки: два приклади подматріци і загальний результат

Тут ми позначили згортку карти ознак X за допомогою матриці ваг W через X\*W, як прийнято позначати згортку в математиці. Якщо розмір згортки буде виражатися парним числом, то в одному з випадків в межах підсумовування нерівність стане суворим; тут більше немає «природного» вибору для центру вікна, і його вибір з чотирьох варіантів може залежати від реалізації в конкретній бібліотеці. Це перетворення має саме тими властивостями, про які ми говорили вище:

– згортка зберігає структуру входу (порядок в одновимірному випадку, взаємне розташування пікселів в двовимірному і т.п.), Так як застосовується до кожної ділянки вхідних даних окремо;

– операція згортки має властивість розрідженості, так як значення кожного нейрона чергового шару залежить тільки від невеликої частки вхідних нейронів (а, наприклад, в повно-нейронної мережі кожен нейрон залежав би від усіх нейронів попереднього шару);

– згортка багаторазово перевикористовує одні і ті ж ваги, так як вони повторно застосовуються до різних ділянок входу.

У більшості випадків після згортки в нейронної мережі слід нелінійність, яку ми можемо записати так як наведено на формулі (2.2):

|  |  |
| --- | --- |
|  | (2.2) |

В якості опції h часто використовують ReLU, особливо в дуже глибоких мережах, але і класичні σ і tanh теж зустрічаються. Докладно зупинятися на типах нелинейностей, що використовуються в згортальних мережах, ми зараз не будемо; наше першочергове завдання – сформувати інтуїції з приводу згортальних мереж.

Розберемо як згортки працюють на практиці, на чисельному прикладі в TensorFlow.

Насамперед створюється згортковий шар на TensorFlow. Згортковий шар повинен брати ковзаючі вікна з вихідного зображення і застосовувати до них одні й ті ж ваги. Згортка у нас двовимірна, тобто «ковзне вікно» – це квадратик розміром в кілька пікселів.

Щоб визначити згортку, спочатку потрібно розібратися з размерностями тензорів. У TensorFlow двовимірні згортки реалізуються за допомогою функції conv2d, яка приймає на вхід тензор згортальних ваг і тензор карт ознак набору зображень. Справа в тому, що вхідні дані для двовимірної згортки в TensorFlow повинні мати чотиривимірну структуру, яка виглядає так:

[розмір батча, высота, ширина, канали].

Наприклад, якщо ми використовуємо міні-батч розміром 32 зображення в кожному і навчаємо мережу на RGB-зображеннях осіб розміром 28 × 28 пікселів, то підсумкова розмірність тензора даних буде [32, 28, 28, 3]: якщо перемножити всі розмірності, вийде, що кожен міні-Батч, що подається на вхід мережі, містить близько 75 тисяч чисел. Кожне зображення при цьому представлено 28×28×3=2352 речовими числами.

А розмірність тензора згортальних ваг визначається розмірами ядра згортки і числом каналів як на вході, так і на виході; виходить знову чотиривимірний тензор, який на цей раз потрібно інтерпретувати в такий спосіб:

[висота, ширина, вхідні канали, вихідні канали]

Наприклад, для фільтра розміром 3×3, що застосовується до того ж самому трьохканальний зображенню і дає на виході 32 карти ознак, ми отримаємо тензор розмірності [3, 3, 3, 32], в ньому буде всього 3×3×3×32=288 ваг. А на виході ці ваги, якщо ми будемо припускати, що в картинку 28×28 пікселів поміщається 26×26 вікон 3×3, перетворять вхідні картинку в 26×26×32=21 632 числа, а це значить, що в повно-мережі для аналогічної операції потрібна була б матриця розміром 28×28×21 632=16 959 488 ваг – різниця в п'ятдесят тисяч разів.

Звичайно, ця різниця виходить виключно через те, що ми перевикористати одні і ті ж ваги багато разів. В реальності виходить, що сама форма операції згортки служить дуже сильним регуляризатора, який виражає ідею про те, що виділення локальних ознак в зображенні не повинно залежати від конкретного місця, де ці ознаки розташовуються. І, безумовно, така величезна різниця в вимагається числі ваг призводить до того, що процес навчання мережі теж сильно спрощується.

На даному етапі може здатися, що і якість підсумкової моделі в разі використання згортальних шарів може постраждати в порівнянні з повнозв'язну шарами. Однак, як показує практика, вірно прямо протилежне: згорткові мережі майже завжди виявляються значно краще повнозв'язних в задачах, пов'язаних з комп'ютерним зором і, взагалі кажучи, обробкою входів, для яких виконується основне припущення згортальних мереж про «локальності»: входи розташовані у вигляді сітки тієї чи іншої розмірності, і ознаки потрібно виділяти з невеликих підмножин входів, розташованих близько до цієї мережі.

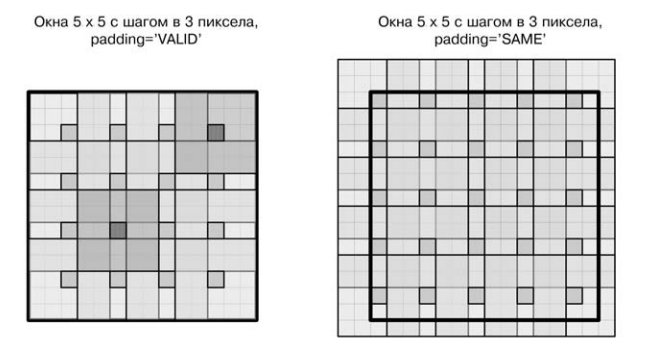
Замінюючи повнозв'язний шар згортки, ми вбиваємо одразу двох зайців:

– додаємо в явному вигляді припущення про те, що ознаки потрібно виділяти локально, а також явно додаємо «геометрію» входу – мережі більше не потрібно самостійно вивчати цю геометрію;

– для локальної мережі, яка покликана виражати ці локальні ознаки, різко збільшуємо обсяг даних на вході: тепер для неї фактично кожна картинка перетворюється в купу маленьких тренувальних прикладів.

Залишилася тільки одна проблема: один згортковий шар ніяк не зможе висловити взаємозв'язок між пікселями, розташованими далеко один від одного. Це дійсно так, і щоб вирішити цю проблему, ми будемо будувати глибокі згорткові мережі; але про це трохи пізніше, а поки повернемося до коду.

Отже, операція згортки в TensorFlow оперує чотиривимірним тензорами, а не звичайними матрицями, тому вхідні дані потрібно привести до відповідних размерностям. Нагадаємо, що перша розмірність тензора x позначає кількість зображень в міні-батче, а остання – число каналів зображення.



а) б)

Рисунок 2.2 – Приклади розташування двовимірних згортальних вікон c центрами в кожному третьому пікселі по обох осях:

а) padding = "VALID", всі вікна повинні вміститися всередині вхідного масиву; б)padding = "SAME", вікна можуть виходити за рамки масиву

Ми можемо задавати padding, щоб говорити, як бути з вікнами, які «вилазять» за межі вхідного масиву. Цей аргумент може приймати два різних значення (рисунок 2.2):

– padding = "VALID" призведе до того, що будуть застосовуватися згортки тільки на тих вікнах, які повністю поміщаються в межах вхідного масиву; це означає, що розмір масиву на виході стане менше, ніж був на вході; наприклад, в нашу картинку розміром 28×28 помістяться тільки 24×24 вікон розміру 5×5, а два пікселя «рамки» не зможуть стати центрами згортальних фільтрів, і наступний шар буде мати розмір 24×24, а не 28×28 (рисунок 2.2, а);

– padding = "SAME" призведе до того, що розмір шару збережеться, а для виходять за межі масиву ми просто доповнимо вхідний масив нулями; тепер на виході розмір зображення не зміниться, а пікселі з «рамки» теж стануть центрами згортальних фільтрів, просто деякі значення у них будуть свідомо нульовими (рисунок 2.2, б).

Історично перші ідеї, схожі на сучасні згортки, з'явилися в уже згадуваній моделі Neocognitron, а вже в практично сучасному вигляді вони були застосовані в кінці 80-х років XX століття. Основним цей напрям став для групи Яна ЛеКуна. Але цікаво, що майже одночасно практично такі ж моделі були розроблені групою японських дослідників під керівництвом Вея Чжана; їх робота отримала назву «нейронна мережа, інваріантна до зрушення» (shift-invariant neural network) і теж неодноразово застосовувалася до розпізнавання образів.

Отже, ми навчилися робити операцію згортки. Після неї можна, як ми вже говорили, застосувати ту чи іншу нелінійну функцію h: вона буде просто застосовуватися до кожного елементу отриманого тензора окремо. Але це ще не все. У класичному згортковому шарі, крім лінійної згортки і наступної за нею нелінійності, є і ще одна операція: субдіскретізація.

Сенс субдіскретізація простий: в згортальних мережах зазвичай виходять із припущення, що наявність або відсутність тієї чи іншої ознаки набагато важливіше, ніж його точні координати. Наприклад, при розпізнаванні осіб згорткової мережею нам набагато важливіше зрозуміти, чи є на фотографії обличчя і чиє, ніж дізнатися, з якого конкретно пікселя воно починається і в якому закінчується. Тому можна дозволити собі «узагальнити» виділяються ознаки, втративши частину інформації про їх місцезнаходження, але зате скоротивши розмірність. Зазвичай в якості операції субдіскретізація до кожної локальної групи нейронів застосовується операція взяття максимуму (max-pooling); така субдіскретізація знову сходить ще до робіт Хьюбела і Визеля, і схоже, що нейрони в зоровій корі надходять саме так. Іноді зустрічаються і інші операції субдіскретізація; згорткові мережі відомі дуже давно, і характерно, що перші конструкції групи ЛеКуна використовували для субдіскретізація взяття середнього, а не максимуму. Дослідження, в яких проводилося порівняння, зазвичай виступали на користь max-pooling. А в роботі докладніше порівняння різних операцій субдіскретізація призвело до того, що в якості оптимальної альтернативи автори запропонували проміжний варіант між максимумом і середнім: не вдаючись у деталі, можна сказати, що вони запропонували брати максимум не по всьому вікну, а по деякому його випадковому подмножеству . Однак саме максимум зустрічається на практиці частіше за все і для більшості практичних завдань дає хороші результати, тому далі, коли ми будемо говорити просто про «субдіскретізація», ми будемо мати на увазі саме maxpooling, і формально станемо визначати субдіскретізація (в тих же позначеннях, що вище) так як наведено у формулі (2.3):

|  |  |
| --- | --- |
|  | (2.3) |

де d – це розмір вікна субдіскретізація.

Як правило, нас цікавить випадок, коли крок субдіскретізація і розмір вікна збігаються, тобто одержувана на вхід матриця ділиться на непересічні вікна, в кожному з яких ми вибираємо максимум; для d=2 ця ситуація проілюстрована на рисунку 2.3, в. Хоча в результаті субдіскретізація дійсно втрачається частина інформації, мережу стає більш стійкою до невеликих трансформацій зображення на зразок зсуву або повороту

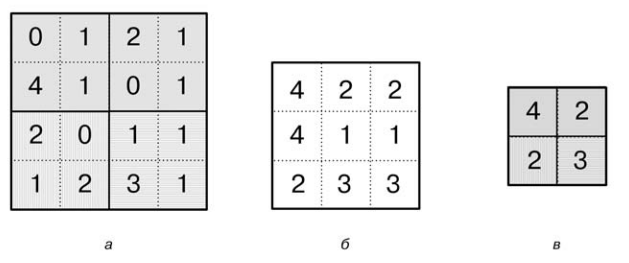


Рисунок 2.3 – Приклад субдіскретізації з вікном розміру 2×2:

а) вихідна матриця; б) матриця після субдіскретізації з кроком 1;

в) матриця після субдіскретізації з кроком 2

Штриховка у вихідній матриці а – відповідає вікнам, за якими береться максимум з кроком 2; в частині в – результат показаний відповідної штрихуванням. Отже, підіб'ємо проміжні підсумки. Стандартний шар згорткової мережі складається з трьох компонентів:

– згортка у вигляді лінійного відображення, що виділяє локальні ознаки;

– нелінійна функція, застосована покомпонентно до результатів згортки;

– субдіскретізація, яка зазвичай скорочує геометричний розмір виходять тензорів.

На рисунку 2.4 зображено такий шар у вигляді, в якому він представлений на стандартних зображеннях згортальних мереж в статтях і посібниках. Окремо малювати нелінійність великого сенсу немає, так що малюють зазвичай дві частини: спочатку згортку, потім Субдіскретізація, вказуючи розмірності. Зверніть увагу, що в порівнянні з «картинкою» на вході розмірність тензора збільшилася: згорткова мережа зазвичай навчає відразу кілька карт ознак на кожному шарі (на рисунку 2.4 таких карт три).

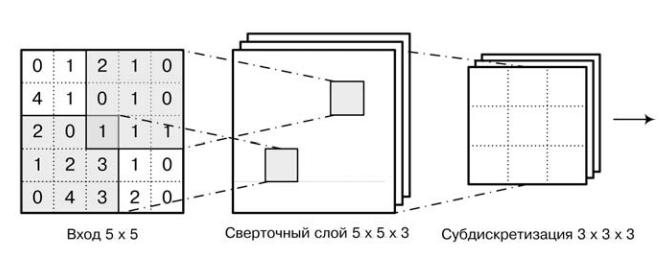


Рисунок 2.4 – Схема одного шару згорткової мережі: згортка, за якою слідує субдіскретізація

Тепер про те, як навчити такі чудові карти ознак. Як будуть проходити градієнти в зворотну сторону через згортковий шар, тобто як, власне, ми будемо навчати згорткову мережу? Припустимо, що ми оптимізуємо деякуфункцію помилки E і вже знаємо її значення на виходах нашого згорткового шару. Щоб провести ітерацію навчання, потрібно зрозуміти, як через них виражаються значення градієнтів функції помилки від ваг.

Через функцію взяття максимуму помилка проходить без змін, шар Субдіскретізація нічого не навчає. Однак він робить проходять по графу обчислення градієнтів розрідженими, адже з усіх елементів вікна Субдіскретізація zli,j приватна похідна ∂E/∂xl+1i,j відноситься тільки до одного, максимальному, а решта отримають нульовий градієнт, і на цьому їх навчання (на даному тренувальному прикладі) можна буде вважати закінченим.

Пропускати через нелінійність ми теж вже добре вміємо: в позначеннях вище як показано у формулі (2.4):

|  |  |
| --- | --- |
|  | (2.4) |

А на згортковому рівні нарешті з'являються ваги, які потрібно вміти навчати. Деяка складність тут полягає в тому, що всі ваги діляться, і кожен бере участь у всіх виходах; так що сума вийде чимала як показано у формулі (2.5):

|  |  |
| --- | --- |
|  | (2.5) |

де індекси i та j пробігають всі елементи картинки на проміжному шарі yli,j, тобто після згортки, але до субдіскретізація.

Для повноти картини залишилося тільки пропустити градієнти на попередній шар як показано у виразі (2.6):

|  |  |
| --- | --- |
|  | (2.6) |

Ось і все: процедуру було адаптовано згідно до зворотного поширення помилки, ця процедура це теж саме, що і зворотний прохід по графу обчислень, для згорткового шару. Зауважимо ще, що зворотний прохід для згортки виявився дуже схожий на, знову ж таки, згортку з тими ж вагами wa,b, тільки замість i+a і j+b тепер i-a і j-b. У разі, коли зображення доповнюється нулями за потребою і розмірності зберігаються, зворотний прохід можна вважати в точності така же згортка, що і в прямому проході, тільки з розгорнутими осями.

## 2.2 Конволюційна нейронна мережа типу ResNet

Можно помітити, що з підвищенням числа шарів згорткова нейронна мережа може почати деградувати – у неї знижується точність на валідаційній безлічі. Так як падає точність і на тренувальній безлічі, можна зробити висновок, що проблема полягає не в перенавчаній мережі.

Було зроблено припущення, що якщо згорткова нейронна мережа досягла своєї межі точності на деякому шарі, то всі наступні шари повинні будуть виродитися в тотожне перетворення, але через складності навчання глибоких мереж цього не відбувається. Для того щоб «допомогти» мережі, було запропоновано ввести пропускне з'єднання (Shortcut Connections), зображене на рисунку 2.5.

Вибір нелінійної активаційної функції проводиться зазвичай виходячи з емпіричних міркувань, хоча найбільш популярними є -функція, розглянута Джоржем Цебенко [1],

(2.7)

з похідною

(2.8)

де так званий «gain parameter», задає форму активационной функції.

Також використовується і функція гіперболічного тангенса

(2.9)

з похідною

. (2.10)

Треба зазначити що, якщо у (2.7) та (2.8) йти до 0 або 1, а в (2.9), (2.10) – до -1 або +1, виникає ефект «зникаючого градієнту».

Саме цей недолік відсутній у так званих функціях типу «випрямовувач», які можна записаси у наступному вигляді

(2.11)

з похідною

(2.12)

Де параметр зазвичай обирається досить довільно. Необхідно заувижити також, що у найбільш популярній активаційній функції цього типу – ReLU .

Природнім узагальненям (2.11) є активаційна функція яку можна записати в наступному вигляді

(2.13)

з похідною

(2.14)

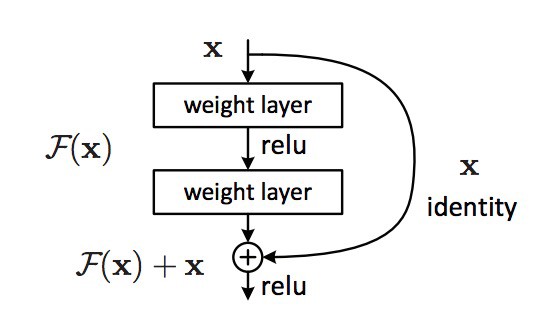


Рисунок 2.5 – Пропускне з'єднання

Пропускне з'єднання пропускають один або кілька шарів і виконують зіставлення ідентифікаторів. Їх виходи додаються до виходів stacked layers. Використовуючи ResNet, можна вирішити безліч проблем, таких як:

– ResNet відносно легко оптимізувати: «прості» мережі (які просто складають шари) показують велику помилку навчання, коли глибина збільшується;

– ResNet дозволяє відносно легко збільшити точність завдяки збільшенню глибини, чого з іншими мережами домогтися складніше.

Реалізація досить проста, але є одне важливе міркування – оскільки вхідні дані Х та X\_shortcut мають відстань більш двох матриць, їх можна додати, тільки якщо вони мають однакову форму. Отже, якщо операції conv2d + batchNorm виконуються таким чином, що форма виводу однакова, то ми можемо просто додати їх, як показано на рисунку 2.6.

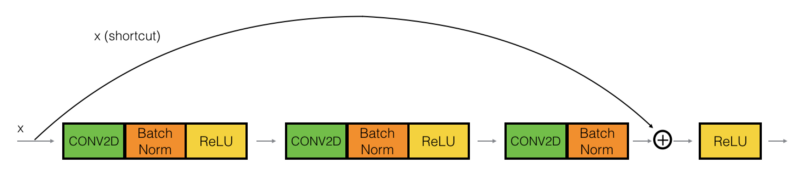


Рисунок 2.6 – Додавання пропускного з’эднання

В іншому випадку, Х\_shortcut проходить через шар згортки, вибраний таким чином, що вихід з нього є таким же розміром, що і вихідний сигнал блоку згортки, як показано на рисунку 2.7.

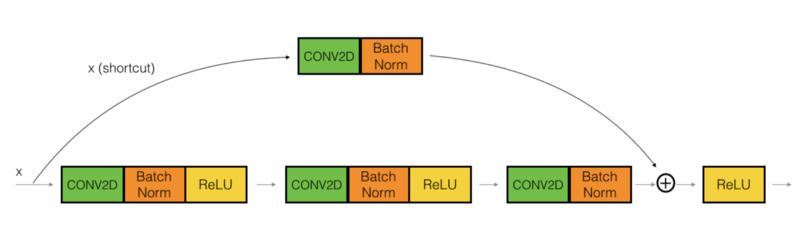


Рисунок 2.7 – Пропускне з’эднання подане через згортковий шар

Важливо відзначити, що пропускне з'єднання застосовується до активації ReLU, як показано на рисунку 2.7. Дослідження показали, що це має найкращі результати.

Чому пропускне з'єднання працює, тут є дві причини:

– воно пом'якшуює проблему зникнення градієнта, дозволяючи цьому альтернативному шляху шляху до градієнта протікати;

– вони дозволяють моделі дізнатися функцію ідентичності, яка гарантує, що більш високий рівень буде виконувати принаймні так само добре, як нижній шар, і не гірше.

2.3 Архітектура нейронної мережі ResUNet

Deep Residual U-Net поєднує ResNet та архітектуру U-net, була запропонована для задачі семантичної сегментації. Складається з двох частин енкодера та декодера з’єднаних блоком мостом, та пропускними з’єднаннями, як показано на рисунку 2.8.

Енкодер складається з блоків ResNet. Кожен блок складається з комбінації шарів пакетного нормування, активаційної функції ReLU та конволюційних шарів з розміром вікна 3×3, як показано на рисунку 2.9. Енкодер приводить вхідне зображення до стислого представлення, яке відображає високорівневі семантичні ознаки.

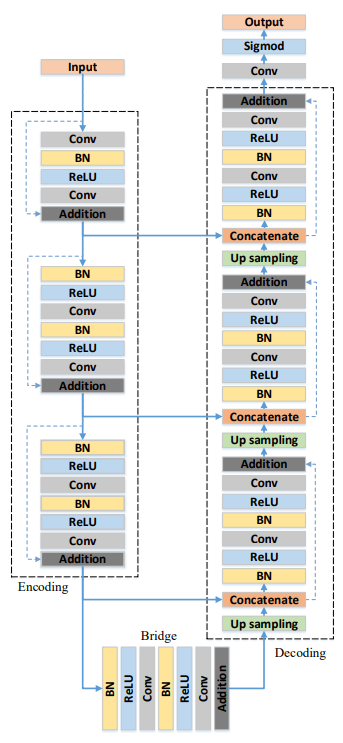


Рисунок 2.8 – Архітектура ResUNet

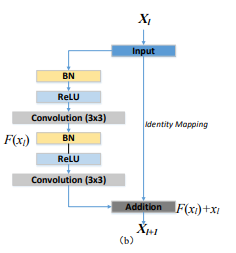


Рисунок 2.9 – Res-блок

Пропускні з'єднання передають виходи блоків енкодера в декодер. Це зумовлює непрямий потік градієнтів до перших шарів, що допомагає передавати градієнти без їхнього погіршення при зворотному поширенні похибки. Такі з’єднання дозволяють використовувати низькорівневу деталізовану інформацію.

Структура моста ідентична з структурою Res-блоків енкодера. Він передає вектор ознак згенерований енкодером на декодер.

Декодер складається з Res-блоків, перед якими виконується транспонована згортка з вікном розмірності 2×2, та конкатинація з тензором переданим пропускними з’єднаннями. Декодер генерує маску семантичної сегментації з низкорівневого представлення вхідного зображення згенерованого енкодером та деталізованої інформації з пропускних з’єднаннь.

До виходу декодера застосовується одновимірна згортка та сігмоїдна функція активації що генерує маску сегментації.

# 3 Імітаційне моделювання згорткової нейронної мережі на основі ResNet

## 3.1 Основні проблеми при класифікації зображень

Якщо виникає проблема нерівномірно наповнених класів доцільно використати наступні методи вирішення цієї проблеми;

– по-перше можна скористатися ансамблем з декількох паралельно працюючих згорткових мереж. Одна згорткова мережа працює з мажоритарними класами, а друга з мінорними;

– по-друге можна штучно добавити екземпляри як дублікати до мінорних класів або видалити навпаки екземпляри з мажоритарних класів;

– по-третє можна додати так звану «штрафну» функцію та збільшити її параметри.

3.2 Трансферне навчання

Трансферне навчання (Transfer Learning) – це використання попередньо навченної на достатньо великому наборі даних моделі (зазвичай використовують ImageNet) в якості відправної точки для моделювання власної моделі.

Використнання трансферного навчання залежить від двох основних параметрів, які треба враховувати при його застосуванні:

– розмір датасету, який буде використано для навчання та подальшого тренування-валідації моделі нейромережі;

–на скільки цей датасет схожий на ImageNet, або на інший великий датасет, на якому буде попердньо навчена модель нейромережі.

Можна виділити чотири основні варіанти, коли використання трансферного навчання бцде доцільним і може покращіти фінальні результати:

1) якщо даних мало, але входовий датасет схожий на Imagenet, то можна взяти виходи з останьогго шару ImageNet і навчити на цих виходах класифікатор;

2) якщо мало даних, але датасети сильно відрізняються, можна взяти виходи не з bottleneck шару, а з якогось внутрішнього шару і навчити на них мережу.

3) якщо датасети різні, але даних багато – можна використовувати ImageNet як ініціалізатор мережі, а потім навчати свою модель як з нуля, але ініціалізувавши її не випадкова, а вже навченими вагами.

4) датасет відносно великий і він схожий на ImageNet.

Мовою згорткових нейронних мереж «схожі» на ImageNet, це набори данних в яких зображення за своєю природою подібні до зображень, які містяться в ImageNet.

331 Програмні засоби та конфігурація комп’ютера

Для моделювання було використано такі програмні засоби:

* Мова програмування Python версії 3.9
* Пакет TensorFlow

Конфігурація комп’ютера:

* Intel Core I5 (4 ядра, 8 потоків, базова тактова частота 2,4 ГГц, максимальна тактова частота в режимі Turbo 4,1 ГГц)
* 16 Гб оперативної пам’яті

3.4 Набір даних

Для навчання та тестування мережі було об’єднано три набори зображень [28, 29, 30]. Набори складаються з пар зображення з синтезованим дощем та зображення без дощу. Дощ був згенерований за допомогою Photoshop 2016 та 2018 року. В результаті було отримано тренувальну вибірку розміром 11854 пар та тестову розміром 2702 пар. У вибірці присутні різні щільності дощу, які відповідають рівню шумів від 5% до 95% та з різними напрямками випадання.

3.5 Попередня обробка даних

Зображення приводяться до розмірності 256×256 пікселів. Проводиться нормалізація методом ділення інтенсивностей пікселів на максимально можливе значення інтенсивності восьмибітного зображення, яке дорівнює 255.

3.6 Архітектура моделі

За основу моделі було взято ResUNet. Енкодер та міст побудовано з шарів попередньо навченої мережі. Декодер складається з конволюційних блоків які мають таку послідовність шарів: шар згортки, пакетна нормалізація, функція активації, шар згортки, пакетна нормалізація, функція активації. Перед кожним конволюційним блоком проводиться транспонована згортка та конкатинація тензора з пропускних з’єднаннь. До виходів енкодера застосовується одновимірна конволюція та функція активації. Оскільки ResUNet була розроблена для задачі сегментації зображень, у джерелі використовується сігмоїдна функція активації на виході мережі. Задача усунення дощу є задачею регресії, тому сігмоїду було замінено на ReLU.

3.7 Навчання моделі

Було начено модель у якій як енкодер використано перший конволюційний шар та настуні три блоки ResNet50[39]. Мостом слугує четвертий блок тієї ж мережі. Декодер складається з чьотироьох блоків описаних у пункті 3.4. Було вибрано функцію активації ReLU та оптимізатор Adam. Цей оптимізатор був вибраний через високу швидкість навчання, для того щоб швидше отримати та оцінити результати, на основі них зробити відповідні зміни в архітектурі та обрати оптимізатор для наступної ітерації. За функцію витрат обрано середньоквадратичну похибку.

Результати навчання:

* MSE для тренувальної вибірки: 0.0021
* MSE для тестової вибірки 0.0093

На основі цих похибок можна зробити висновок про перенавчання.

Для покращення результатів мережа була зменшена. Четвертий Res-блок ResNet50, який був мостом та перший блок декодера були усунуті. Останній блок енкодера після змін використовується як міст. Функція активації була замінена на GELU, рисунок 3.1. Оскільки якщо під час навчання ReLU отримає значення в проміжку від -∞ до 0 включно то, оскільки похідна в цьому проміжку дорівнює нулю, градієнт який передається наступному шару завжди буде дорівнювати нулю.

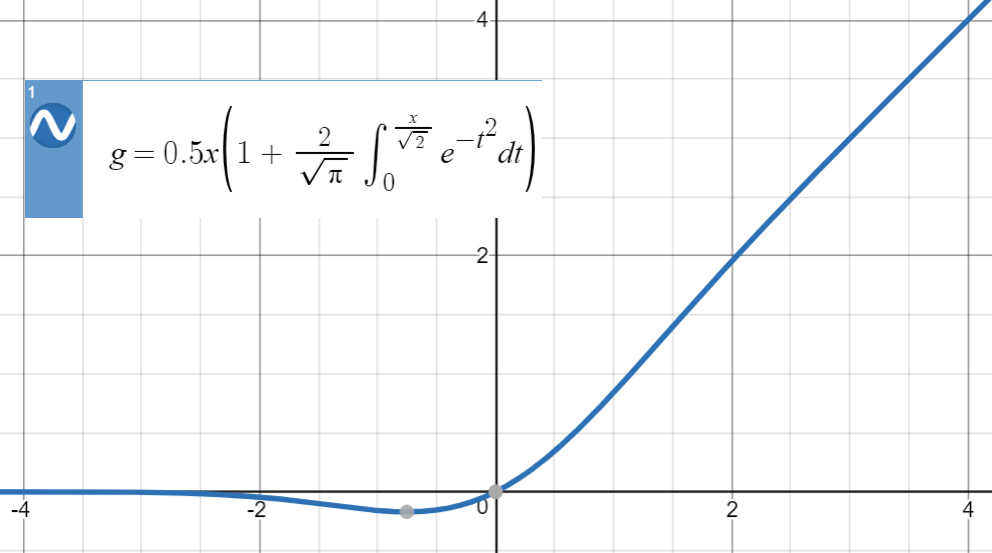


Рисунок 3.1 - Лінійна одиниця похибки Гауса (GELU)

Було використано оптимізатор SDG, порівняно з Adam повільніше досягає конвергенції, але має кращі генералізаційні якості. Для пр швидшення навчання було використано імпульс (анг. momentum).

Результати навчання:

* MSE для тренувальної вибірки: 0.0033
* MSE для тестової вибірки: 0.0028

3.9 Аналіз отриманої якості моделі та порівняння з існуючими моделями

Візуальний аналіз зображень показує незначну різницю між згенерованими і істинними зображеннями, рисунки 3.3, 3.4



Рисунок 3.2 – Зображення з синтетичним дощем



Рисунок 3.3 – Зображення згенероване мережею



Рисунок 3.4 – Істинне зображення

Деякі регіони зображення темніші на згенерованому зображенні ніж на істинному, рисунок 3.5.

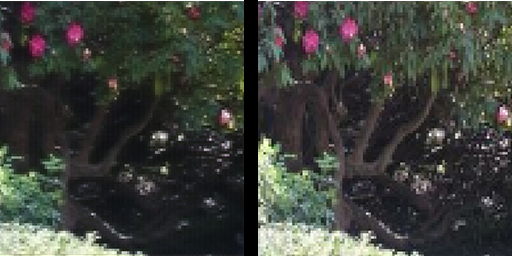


Рисунок 3.5 – Порівняння деталей 1, ліворуч згенероване мережею, праворуч істинне зображення

На зображеннях з граничною щільністю дощу залишається частина шумів, рисунок 3.6.



Рисунок 3.6 – порівняння деталей 2, ліворуч згенероване мережею, праворуч істинне зображення.

Проведено порівняння з існуючими рішннями за допомогою SSIM метрики, таблиці 3.1 та 3.2.

Таблиця 3.1 – Порівняння моделей на наборі даних Test1[30]

|  |  |
| --- | --- |
| Модель | SSIM |
| GMM[36] | 0.7896 |
| DSC[32] | 0.8352 |
| JORDER[33] | 0.8622 |
| JBO[37] | 0.8356 |
| DID-MDN[30] | 0.9087 |
| **ResUNet** | **0.8746** |

Таблиця 3.3 – Порівняння моделей на наборі даних Rain1400[28]

|  |  |
| --- | --- |
| Модель | SSIM |
| LP[34] | 0.836 |
| DSC[32] | 0.799 |
| JCAS[35] | 0.819 |
| JORDER[33] | 0.853 |
| DID-MDN[30] | 0.869 |
| **ResUNet** | **0.871** |

На наборі даних Test1[30], таблиця 3.1, запропонована модель є другою по точності з порівняних. На наборі даних Rain1400[28] модель показує найкращі результати серед порівняних. При цьому різниця між значеннями метрик для двох вибірок нижча ніж у інших моделей представлених в обох таблицях, що вказує на вищу робастність.

# 

# Висновки

В кваліфікаційній роботі бакалавра представлені результати, що є відповідно до поставленої мети рішенням актуальної задачі розробки згорткової нейронної мережі для опрацювання зображень за умов поганої видимості. Розроблена модель згортової глибинної нейромережі типу ResUNet дозволяє опрацьовувати зображення та покращувати їх якість.

Проведені дослідження дозволили зробити наступні висновки.

1. Проаналізовано стан проблеми розпiзнавання образiв і розглянуті існуючі підходи до її вирішення. Проаналізовано стан сучасної теорії ШНМ, призначених для вирішення задач обробки даних в тому числі і задач інтелектуального аналізу даних, а також розглянуто основні підходи до її реалізації.

2. Було проведено навчання моделі згорткової глибокої нейромережі типу ResUNet, яка опрацьовує зображення, що були зняті за умов поганої видимості;

3. Було проведено аналіз отриманих результатів та «тонке» налаштування моделі;

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature.* 2015. № 521. P. 436–444.

2. Krizhevsky A., Sutskever I., Hinton G. Imagenet classification with deep convolutional neural networks. *Proc. Adv. Neural Inf. Process. Syst*. 2012. № 12. Р. 1097–1105.

3. Schalkoff R. Artificial Neural Networks. N.Y.: The McGraw–Hill Comp., Inc., 1997. 422 p.

4. Haykin S. Neural Networks. A Comprehensive Foundation. Upper Saddle River, N.J.: Prentice Hall, Inc., 1999. 842 p.

5. Poggio T., Girosi F. A Theory of Networks for Approximation and Learning. Massachussetts Institute of Technology, 1994. 63 p.

6. Bishop, Christopher M. Pattern Recognition and Machine Learning. Springer. 2006.

7. Howard W.R. Pattern Recognition and Machine Learning. 2007.

8. Chiswell Ian. *Mathematical logic*. Oxford University Press, 2007. p. 34.

9. Fukushima K. Neocognitron. *Scholarpedia*. 2007. 2(1). Р. 1717.

10. Matusugu Masakazu, Katsuhiko Mori, Yusuke Mitari, Yuji Kaneda (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. P. 555–559.

11. Персональна сторінка Ян ЛеКунна у Фейсбук : веб-сайт. URL: https://www.facebook.com/yann.lecun/posts/10152820758292143 (дата звернення 10. 11.2020).

12. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens. *Rethinking the Inception Architecture for Computer Vision*. 2015. №11. URL: https://arxiv.org/pdf/1512.00567v3.pdf (дата звернення 10. 11.2020).

13. Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.* 2016. №2. URL:https://arxiv.org/pdf/1602.07261v1.pdf (дата звернення 15. 11.2020).

14. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Identity Mappings in Deep Residual Networks.* 2016. №4. URL: https://arxiv.org/pdf/1603.05027v2.pdf (дата звернення 8. 11.2020).

15. Qianli Liao, Tomaso Poggio. *Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex.* 2016. №4. URL: https://arxiv.org/pdf/1604.03640.pdf (дата звернення 7. 11.2020).

16. Andreas Veit, Michael Wilber, Serge Belongie. *Residual Networks are Exponential Ensembles of Relatively Shallow Networks.* 2016. №5. URL: https://arxiv.org/pdf/1605.06431v1.pdf (дата звернення 7. 11.2020).

17. Mohammadreza Amirian, Friedhelm Schwenker. IEEE: 2020, 123097 p. Radial Basis Function Networks for Convolutional Neural Networks to Learn Similarity Distance Metric and Improve Interpretability. URL: https://ieeexplore.ieee.org/abstract/document/9133368/figures#figures (дата звернення 10. 11.2020).

18. Poggio T., Girosi F., Networks for approximation and learning, Proc. IEEE, vol. 78, no. 9, pp. 1481-1497, Sep. 1990.

19. LeCun Y., Bottou L., Bengio Y., Haffner P., Gradient-based learning applied to document recognition, Proc. IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

20. Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., ImageNet large scale visual recognition challenge, Int. J. Comput. Vis., vol. 115, no. 3, pp. 211-252, Dec. 2015.

21. Masi I., Wu Y., Hassner T., Natarajan P., Deep face recognition: A survey, Proc. 31st SIBGRAPI Conf. Graph. Patterns Images (SIBGRAPI), pp. 471-478, Oct. 2018.

22. Hu J., Lu J., Tan Y.-P., Deep transfer metric learning, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 325-333, Jun. 2015.

23. Wang J., Zhou F., Wen S., Liu X., Lin Y., Deep metric learning with angular loss, Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 2593-2601, Oct. 2017.

24. Cheng G., Yang C., Yao X., Guo L., Han J., When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs, IEEE Trans. Geosci. Remote Sens., vol. 56, no. 5, pp. 2811-2821, May 2018.

25. Tan M., Le Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. Proc. 36th Int. Conf. Mach. Learn., 2019. pp. 6105-6114.

26. Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the inception architecture for computer vision. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016. pp. 2818-2826.

27. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016. 836 p.

28. X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding and J. Paisley. Removing Rain from Single Images via a Deep Detail Network, CVPR, 2017.

29. X. Fu, J. Huang, X. Ding, Y. Liao and J. Paisley. Clearing the Skies: A deep network architecture for single-image rain removal, IEEE Transactions on Image Processing, vol. 26, no. 6, pp. 2944-2956, 2017.

30. He Zhang Vishwanath Sindagi, Vishal M. Patel Density-aware Single Image De-raining using a Multi-stream Dense Network, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 695-704

31. Z. Zhang, Q. Liu and Y. Wang, "Road Extraction by Deep Residual U-Net," in IEEE Geoscience and Remote Sensing Letters*,* vol. 15, no. 5, pp. 749-753, May 2018

32. Yu Luo, Yong Xu, and Hui Ji. Removing rain from a single image via discriminative sparse coding. In Proceedings of the IEEE International Conference on Computer Vision, pages 3397–3405, 2015.

33. Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1357–1366, 2017.

34. Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2736–2744, 2016

35. Shuhang Gu, Deyu Meng, Wangmeng Zuo, and Lei Zhang. Joint convolutional analysis and synthesis sparse representation for single image layer separation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1708–1716, 2017

36. Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown. Rain streak removal using layer priors. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2736–2744, June 2016

37. L. Zhu, C.-W. Fu, D. Lischinski, and P.-A. Heng. Joint bilayer optimization for single-image rain streak removal. In Proceedings of the IEEE international conference on computer vision, pages 2526–2534, 2017.

38. Fu, Jun, Jianfeng Xu, Kazuyuki Tasaka and Zhibo Chen. Residual Squeeze-and-Excitation Network for Fast Image Deraining. ArXiv abs/2006.00757 (2020): n. pag.

39. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

40. Hendrycks, Dan and Gimpel, Kevin, Gaussian Error Linear Units (GELUs), 2016. URL: https://arxiv.org/abs/1606.08415

ДОДАТОК А

Вихідний код програми

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, MaxPool2D, Conv2DTranspose, Concatenate, Input

from tensorflow.keras.models import Model

from tensorflow.keras.applications import ResNet50

import matplotlib.pyplot as plt

def conv\_block(input, num\_filters):

x = Conv2D(num\_filters, 3, padding="same")(input)

x = BatchNormalization()(x)

#x = keras.layers.LeakyReLU()(x)

#x = Activation('relu')(x)

x = keras.activations.gelu()(x)

x = Conv2D(num\_filters, 3, padding="same")(x)

x = BatchNormalization()(x)

#x = keras.layers.LeakyReLU()(x)

#x = Activation('relu')(x)

x = keras.activations.gelu()(x)

return x

def decoder\_block(input, skip\_features, num\_filters):

x = Conv2DTranspose(num\_filters, (2, 2), strides=2, padding="same")(input)

x = Concatenate()([x, skip\_features])

x = conv\_block(x, num\_filters)

return x

def build\_resnet50\_unet(input\_shape):

""" Input """

inputs = Input(input\_shape)

""" Pre-trained ResNet50 Model """

resnet50 = ResNet50(include\_top=False, weights="imagenet", input\_tensor=inputs)

""" Encoder """

s1 = resnet50.get\_layer("input\_1").output ## (512 x 512)

s2 = resnet50.get\_layer("conv1\_relu").output ## (256 x 256)

s3 = resnet50.get\_layer("conv2\_block3\_out").output ## (128 x 128)

""" Bridge """

b1 = resnet50.get\_layer("conv3\_block4\_out").output ## (64 x 64)

""" Decoder """

d2 = decoder\_block(b1, s3, 256) ## (128 x 128)

d3 = decoder\_block(d2, s2, 128) ## (256 x 256)

d4 = decoder\_block(d3, s1, 64) ## (512 x 512)

""" Output """

outputs = Conv2D(3, 1, padding="same", activation="relu")(d4)

model = Model(inputs, outputs, name="ResNet50\_U-Net")

return model