Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

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

Дослідження та розробка методів покращення зображень.
.
.
(тема)

Виконав:
студент 2 курсу, групи СШІм-18-1
Михайлов В.С.
(прізвище, ініціали)
Спеціальність 122 – Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо -наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва освітньої програми)
Керівник доц. Магдаліна І.В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

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

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

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Михайлову Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка методів покращення зображень

затверджена наказом по університету від 04 листопада 2019 р. № 1623Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 грудня 2019 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки та дослідження методів покращення зображень

4. Перелік питань, що потрібно опрацювати в роботі
1 Аналіз розвитку та проблем покращення зображень
2 Аналіз методів та алгоритмів покращення зображень
3 Аналіз нейронної мережі GAN
4 Розробка програмного додатку покращення вхідних зображень
5 Порівняння результатів роботи алгоритмів і моделей ШНМ за обраних метрик

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _Рисунок 1 – Схема кодування JPEG, Рисунок 2 – Схема декодування JPEG, Рисунок 3 – Архітектура SRCNN, Рисунок 4 – Архітектура генератора, Рисунок 5 – Архітектура дискримінатора, Рисунок 6 – Динаміка навчання SRGAN, Рисунок 7 – Результати роботи алгоритмів._____
_____
_____
_____
_____
_____
_____

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Магдаліна І.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	04.11.2019	виконано
2	Аналіз предметної області	10.11.2019	виконано
3	Постановка завдання та узгодження з керівником	15.11.2019	виконано
4	Дослідження методів покращення зображень	20.11.2019	виконано
5	Дослідження алгоритмів покращення зображень	22.11.2019	виконано
6	Розробка програмного забезпечення	25.11.2019	виконано
7	Написання пояснювальної записки	30.11.2019	виконано
8	Попередній захист	16.12.2019	виконано
9	Захист перед ЕК	19.12.2019	виконано

Дата видачі завдання 04.11.2019 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 91 с., 10 рис., 3 табл., 9 дод., 29 джерел.

АЛГОРИТМ JPEG, ГЕНЕРАТИВНО ЗМАГАЛЬНА МЕРЕЖА, ІНТЕРПОЛЯЦІЯ, НЕЙРОННІ МЕРЕЖІ, ПОКРАЩЕННЯ ЗОБРАЖЕНЬ

Метою роботи є дослідження методів та алгоритмів покращення зображень.

Об'єктом дослідження є системи та алгоритми які використовуются для покращення вхідних зображень.

У даній роботі запропоновано використання методу покращення вхідних зображень що ґрунтується на основі нейронної мережі GAN. Перевага цього методу в тому що він може використовуватися для поліпшення якості нечітких або частково зіпсованих зоб'ржень. Проведено дослідження результатів покращення зображень на базі нейронної мережі GAN.

У результаті роботи здійснена програмна реалізація системи для покращення вхідних зображень з використанням нейронної мережі GAN.

РЕФЕРАТ

Пояснительная записка: 91 с., 10 рис., 3 табл., 9 прил., 29 источников.

АЛГОРИТМ JPEG, ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНАЯ СЕТЬ, ИНТЕРПОЛЯЦИЯ, НЕЙРОННЫЕ СЕТИ, УЛУЧШЕНИЕ ИЗОБРАЖЕНИЙ

Целью работы является исследование методов и алгоритмов улучшения изображений.

Объектом исследования являются системы и алгоритмы, которые используются для улучшения входных изображений.

В данной работе предложено использование метода улучшения входных изображений основанного на нейронной сети GAN. Преимущество этого метода в том, что он может использоваться для улучшения качества нечетких или частично испорченных изображений. Проведено исследование результатов улучшения изображений на базе нейронной сети GAN.

В результате работы осуществлена программная реализация системы для улучшения входных изображений с использованием нейронной сети GAN.

ABSTRACT

Explanatory note: 91 pages, 10 figures, 3 tables, 29 sources, 9 supplement.

ALGORITHM JPEG, GENERATIVE ADVERSARIAL NETWORK, IMAGES IMPROVEMENT, INTERPOLATION, NEURAL NETWORKS

Objective of the work is to study methods and algorithms for images improvement.

Objects of the research are systems and algorithms used to improve input images.

In this work, it was proposed to use a method of GAN neural network for input images improvement. The main advantage of this method is that it can be used to improve the quality of fuzzy or partially corrupted images. The research was conducted on the results of image enhancement based on the GAN neural network.

As a result of the work, was made a software implementation of the system for input images improvement using the GAN neural network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШНМ – штучна нейронна мережа;

DCCI – Directional Cubic Convolution Interpolation – інтерполяція спрямованої кубічної згортки;

ESPCN – Efficient Sub-Pixel Convolutional Neural Network – конвультивна субпіксельна нейронна мережа;

MSE – Mean Squared Error – середньоквадратична помилка;

PSNR – Peak Signal-to-Noise Ratio – пікове відношення сигналу до шуму;

ReLU – Rectified Linear Unit – «випрямляч»;

SRCNN – Super-Resolution Convolutional Neural Network – конволюційна нейронна мережа суперроздільності;

SRGAN – Super-Resolution Generative Adversarial Network – генеративно змагальна мережа;

SSIM – Structural Similarity – індекс структурної подібності;

VDSR – Very Deep Convolutional Network – глибока згорткова мережа.

ВСТУП

В сучасній науці, та й в усьому світі, напевно, мало знайдеться області роботи людини, де б не використовувалося зображення. У поточний вік, де світом правлять комп'ютери, стрімко розвивається область цифрової обробки зображень.

Дана проблема була розглянута в статті Захарова Р.К. «Методи підвищення якості зображень в задачах розпізнавання».

Багато галузей техніки, що мають відношення до отримання, обробки, зберігання та передачі інформації, в даний час орієнтуються на розвиток систем, в яких інформація може надаватися у вигляді зображень. Зображення, яке можна розглядати як двовимірний сигнал, є значно більш ємним носієм інформації, ніж звичайний одновимірний, найчастіше тимчасової, сигнал. Разом з тим, рішення наукових і інженерних задач при роботі з візуальними даними вимагає особливих підходів, що спираються на знання специфіки зображень, оскільки традиційні способи обробки та аналізу одновимірних сигналів мало придатні в цих випадках. Однією з основних проблем, що виникають при обробці зображень, є поліпшення якості зображення, зокрема, підвищення розрізнення окремих фрагментів. До причин, що знижує якість зображень, можна віднести:

- технічні характеристики яскравості перешкоди шумового характеру;
- недостатня або надмірна освітленість об'єктів зйомки;
- відсутність різкості при отриманні зображення;
- занадто дрібні розміри деталей, які необхідно розрізнити.

Основною метою комп'ютерної обробки зображень є знаходження методів, результат роботи яких виявився б більш підходящим з точки зору конкретного застосування.

1 ВИДИ І ФОРМАТИ ЗОБРАЖЕНЬ

Зображення – важливе поняття, що стоїть в одному ряду з чином, знаком і ін. Подібно моделі, зображення лише порівняно недавно стало предметом наукового вивчення і викликає зростаючий інтерес. У той же час використання зображень в різних сферах діяльності повертає нас до далекого минулого людства.

Уже в епоху кам'яного віку ми зустрічаємося з фігурками і печерними і наскельними малюнками, які зображують звірів і людей. В нашу епоху бурхливий розвиток технічних способів отримання зображень (фото-кіно зображення) і передача їх на великі відстані (телебачення, інтернет) призвело до надзвичайно широкого поширення їх у всіх сферах діяльності і в побуті. Це спонукало деяких соціологів проголосити народження «нової ери» – ери зображень, що змінила «книжкову еру». Важливе наукове і практичне значення набуває питання про розпізнавання зображень технічними пристроями.

Проблема зображення вплітається в вивчення процесів моделювання, проблеми «наочності» в сучасній фізиці.

Потреба в спеціальному дослідженні цього поняття відчувають естетична наука і мистецтвознавство. Особливо гостро постало це питання в зв'язку з розвитком абстракціонізму та інших авангардистських напрямків в образотворчих мистецтвах.

Які ж основні ознаки зображення взагалі? Це предмети (системи), що володіють такими властивостями: вони повинні бути матеріальні, цілісні, безперервні, мати відоме схожість з іншим предметом. Ступінь подібності може бути виражена – у всякому разі для більшості зображень – в понятті подібності, яке визначається як подібність форми (геометричної, динамічною, кінематичної) при розходженні в інших відносинах (величини і ін.).

Предмети (системи), що володіють цими ознаками виступають як зображення, якщо функціонують як знаки. Під знаком розуміється будь-яке матеріальне явище, що створюється або використовується людиною з метою передати з його допомогою будь-яку понятійну або емоційну інформацію. Існує кілька основних різновидів знаків, використовуваних для комунікації: умовні, виразні, словесні і образотворчі, або зображення. Наукові, навчальні фільми, телебачення, схеми, креслення, моделі, образотворчі дорожні знаки, фотозображення, художні картини – далеко не повний перелік використання образотворчих знаків. Їх широке використання в мистецтві дає привід деяким авторам вважати образотворчий знак специфічним естетичним знаком.

Як зображень можуть виступати як природні явища (сліди, відбитки), так і штучні, створені людиною. Серед останніх – продукти автоматичної роботи апаратів – фото-кіноапаратом і ін. Зображення залежить як від зображуваного об'єкта, так і від людини, суб'єкта, від тієї мети, яку ставить перед собою людина, створюючи або відтворюючи образотворчий знак. Цим і визначається специфіка образотворчих знаків в науці, мистецтві, техніці і т.д. Вони можуть функціонувати різним способом: характеризувати, оцінювати, наказувати і т.п.

Відповідно зображення можуть бути науковими, міфологічними, поетичними, релігійними і можуть служити самим різним індивідуальним і соціальним цілям (інформативним, естетичним і т.д.). Сильна сторона образотворчого знака – в його здатності представляти для споглядання то, що він позначає, його «картинність», наочна слабкість – в обмеженні об'єктів позначення лише тим, що подібно до нього.

Основные характеристики растрового изображения – размер и глубина цвета.

Размер изображения в пикселях – это количество строк и столбцов матрицы, использующихся для хранения изображения.

Позначати, як правило, не є кінцевою функцією зображень. Це скоріше основну властивість, без якого воно не існує. Кінцеві цілі можуть бути різними. У тому випадку, коли зображення служить цілям пізнання, воно виступає як образотворча або иконическая модель. Вона призначена дати точну інформацію про об'єкт, не переслідуючи при цьому завдання домогтися будь-якого емоційного ефекту і оцінного ставлення до інформації з боку інтерпретатора. Навпаки, образотворчий знак у мистецтві – скажімо портрет, пейзаж, скульптура тощо – не тільки передає інформацію про об'єкт, але обов'язково сприяє тому, щоб у глядача виникло певне оцінне, емоційно-забарвлене відношення як до зображуваного, так і до самого образотворчого знаку, причому ставлення до самого знаку має бути почуттям естетичного задоволення. Художність самих образотворчих «кодів» в мистецтві виступає як найважливіший і необхідний компонент їх значення, їх сенсу, як необхідне і суттєве «доданок» художнього образу. «Код» і «повідомлення» в мистецтві нерозривні. З цим пов'язані труднощі формалізації мови мистецтва, його адекватної переводимости з однієї системи знаків на іншу, особливо на мову точних наукових термінів.

1.1 Види і формати цифрових зображень

Цифрове зображення – масив даних, отриманий шляхом дискретизації (аналого-цифрового перетворення) оригіналу. Будучи закодованим за допомогою особливого алгоритму і записаним на носій, цей масив даних стає файлом [1].

Цифрові зображення за способом дискретизації оригіналу поділяються на растрові, векторні і змішаного типу.

Растрове зображення складається з серії точок і пікселів. Кожен піксель утворений одиницями і нулями, або бітами. Кількість бітів, який комп'ютер використовує для того, щоб створити кожен піксель,

відображається в насиченості кольору зображення. Растрове зображення може складатися мінімум з одного біта, максимум з 256 бітів. Майте на увазі, що чим більше буде число бітів, тим більше буде розмір файлу. Збільшення растрового зображення тягне за собою його спотворення і обривання, відоме як розкладання зображення на елементи (пікселація).

Типи формату растрового зображення включають в себе: формат графічного обміну GIF, формат JPEG (формат зберігання графічних файлів, стислих за допомогою алгоритму JPEG), побітовое зображення Bitmap, формати MAC, PSP і TIFF.

Найосновнішими растровими форматами є GIF і JPEG.

До растрових зображень відносяться двомірні масиви даних (матриці пікселів), кожен елемент яких представляє ділянку оригіналу з усередненим колірним показником.

Растрові зображення отримують двома способами. Перший – сканування оригіналу – проводиться за допомогою особливого пристрою – сканера – в якому кожен оптичний елемент ПЗС-лінійки (або ПЗС-матриці) зчитує яскравості і колірні характеристики оригіналу. Ці характеристики перетворюються в двійковий код кольору і надсилаються в осередку двомірного масиву даних (матриці пікселів). Другий спосіб отримання растрового зображення – проектування оригіналу на ПЗС-матрицю через систему лінз (об'єктив). Цей спосіб реєстрового аналого-цифрового перетворення характерний для цифрових фотоапаратів і відеокамер.

Дозвіл цифрового зображення можна довільно змінювати, змінюючи фізичний розмір картинки при друку, при цьому розмір матриці пікселів залишатиметься незмінним.

Глибина кольору – це характеристика, що визначає якість відтворення кольору, кількість відтінків, які можуть відображати елементи матриці пікселів.

Кожен елемент масиву даних (матриці) являє собою число в двійковій системі числення. Його розмірність визначається в бітах.

Глибина кольору – це кількість біт на піксель зображення. Зображення з глибиною кольору 16 біт / піксель може відтворювати 65.535 квітів, а 24 біт / піксель дозволяють отримати вже 16.777.215 відтінків, що цілком достатньо для поліграфічного виробництва.

Растрові зображення використовуються у всіх випадках, коли необхідно відтворити аналоговий оригінал, будь то фотографія, малюнок, складний елемент оформлення, який нераціонально переводити в вектори.

Другим видом цифрових зображень є векторні зображення.

Векторне зображення, зроблене для певної програми, є чимось більшим, ніж просто набір кольорових крапок. Це векторні дані, що зберігаються в математичних форматах, які можуть створювати будь-яку кількість обрисів, які в свою чергу можна відредагувати без втрати якості. Векторні зображення є універсальними і використовуються в контексті створення двомірної комп'ютерної графіки.

Зразки зображень векторного формату включають: SVG (масштабована векторна графіка), SWF (формат Flash Player), CDR (векторний формат файлів програми CorelDraw) і EMF (розширений метафайл).

Зображення векторного формату створені в векторній графіці малюють програм або навіть за допомогою Інструментів Office.

Найменшими елементами векторного зображення є вектор і крива Безьє. Вектор в комп'ютерній графіці – це відрізок, що з'єднує дві точки з заданими координатами. Основним елементом, що управляє кривої Безьє є вузол (node), також званий контрольної точкою (CP, control point) або контрольної вершиною (CV, control vertex). Ступінь кривизни лінії визначаються координатами вузла і двох керуючих точок.

Контур зображення в цифровому вигляді являє собою масив даних, що містить координати контрольних і керуючих точок, а також характеристики кривої в цілому – її товщину, колір, напрямок, а якщо крива замкнута - то і колір і тип заливки.

Примітиви є прості геометричні форми, які в масиві даних кодуються цілком, без поділу на криві Безьє і вектора, умовним кодом тієї чи іншої геометричної фігури, а також кодами розміру фігури, її координатами, кодами типу і кольору заливки фігури, товщини і кольору контура та інших характеристик.

Інформація про текст, що становить частину зображення, як правило, буває представлена у вигляді ASCII кодів символів, що супроводжуються цифровою інформацією про візуальних характеристиках текстового блоку – гарнітурі шрифту, накреслення, колір контура і заливки, способі заливки, методі вирівнювання тексту в блоці і т.д.

Векторні зображення отримують двома способами – шляхом ручного трасування оригіналу і шляхом автоматичного трасування.

При ручному трасуванні художник або дизайнер фактично «з нуля» створює зображення, як би «обводячи» наявні контури, за допомогою графічного редактора задаючи вектора, криві Безьє і графічні примітиви.

Автоматичне трасування оригіналу проводиться за допомогою програмного забезпечення, яке за допомогою інтелектуальних алгоритмів розпізнає контури оригіналу, представленого растрових зображенням, і на основі отриманої інформації відтворює лінії, заливки та ін. Таким чином, щоб з них склалося векторне зображення, максимально близьке до оригіналу.

Основна перевага векторного зображення – це можливість масштабування без втрати якості.

Ще одним плюсом векторних зображень є порівняно невеликий розмір файлів, що їх містять. Це робить зручною передачу векторних зображень по електронних каналах зв'язку. Головний недолік векторних зображень – це те, що вони майже завжди відтворюють оригінал в спрощеному вигляді. Деякі деталі оригіналу буває неможливо відтворити в векторному зображенні.

1.1.1 Детальний огляд формату JPEG

JPEG – це стандарт стиснення зображень, який був створений об'єднаної групою експертів по фотографії (Joint Photographic Experts Group). Для задоволення різних потреб стандарт JPEG включає в себе два основні методи стиснення, кожен з яких має різні режими роботи. Метод дискретного косинусного перетворення (ДКП) застосовується для стиснення з «втратами» і прогностичний метод застосовується для стиснення без втрат.

1.1.2 Опис алгоритму стиснення JPEG

Для алгоритму, який повинен був стати стандартом стиснення зображень, виставлялися такі вимоги:

- алгоритм повинен бути одним з кращих зі співвідношення ступеня стиснення до якості зображення для зображень різної якості. Також алгоритм повинен бути параметрізіруемим, щоб користувач міг встановити бажане співвідношення ступеня стиснення до якості;

- алгоритм повинен бути придатним до будь-якого типу цифрового зображення (тобто не повинно бути обмежень за розмірами зображення, по колірних просторів і т.д.) і не обмежуватися певними класами зображень з обмеженнями на діапазон кольорів або статистичні властивості;

- алгоритм повинен мати прийнятну обчислювальну складність, як для створення ефективних програмних реалізацій на різних процесорах, так і для створення ефективних апаратних реалізацій для задач, що вимагають високої продуктивності.

Алгоритм повинен мати такі режими роботи:

- послідовне подання даних: послідовний обхід кодованого зображення по блоках зліва направо, зверху вниз;

- прогресивне уявлення даних: алгоритм видає набір сканів, кожен з яких описує зображення з все більшим ступенем деталізації. Це дозволяє використовувати даний алгоритм, в тому випадку, якщо швидкість з'єднання при передачі зображення вкрай мала – клієнт може отримати уявлення про зображення навіть після передачі невеликої частини файлу;

- стиснення без втрат: для точного відновлення оригінального зображення.

JPEG стандарт містить «режими роботи», описаних раніше. Для кожного режима вказується один або кілька окремих кодеків. Кодеки в режимі відрізняються в залежності від якості зразків вихідного зображення, які вони можуть обробляти, або від методу ентропії кодування. Тепер розглянемо базовий режим роботи алгоритму на прикладі зображення у відтінках сірого.

На рисунку 1.1 і рисунку 1.2 зображені основні кроки алгоритмів кодування і декодування.

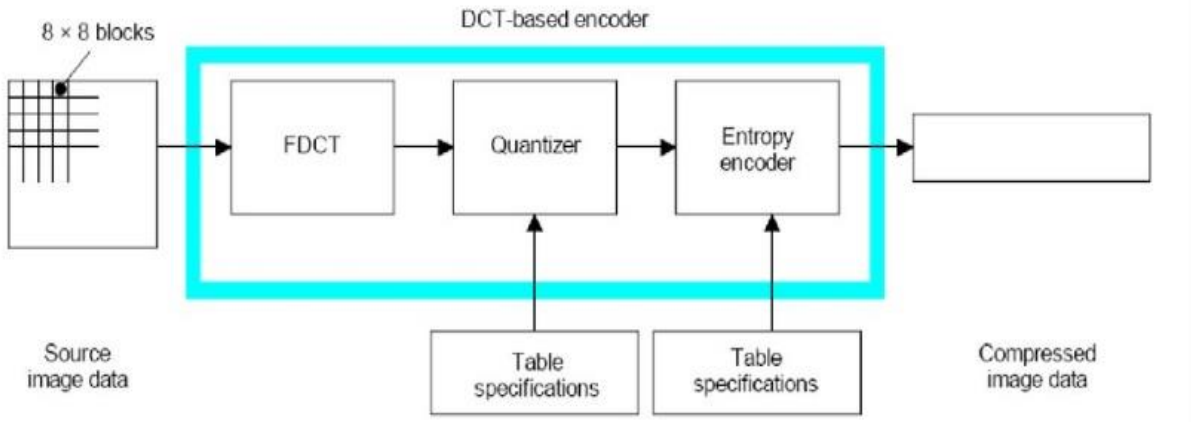


Рисунок 1.1 – Схема кодування JPEG

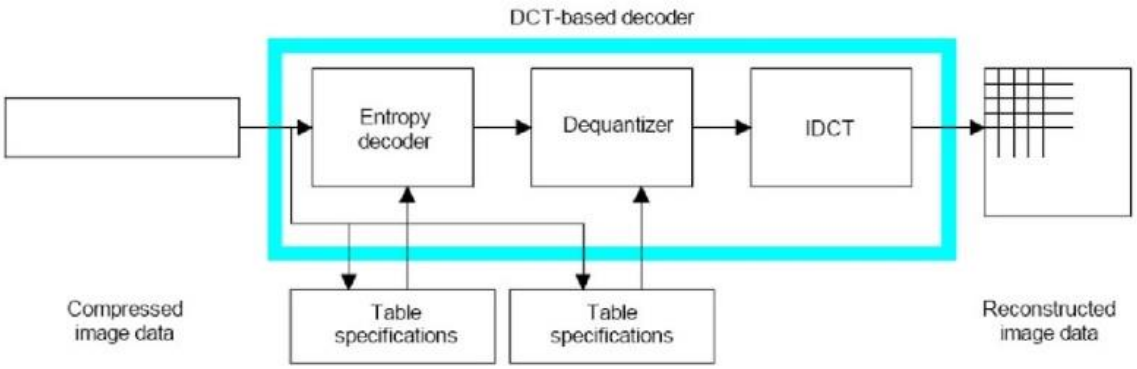
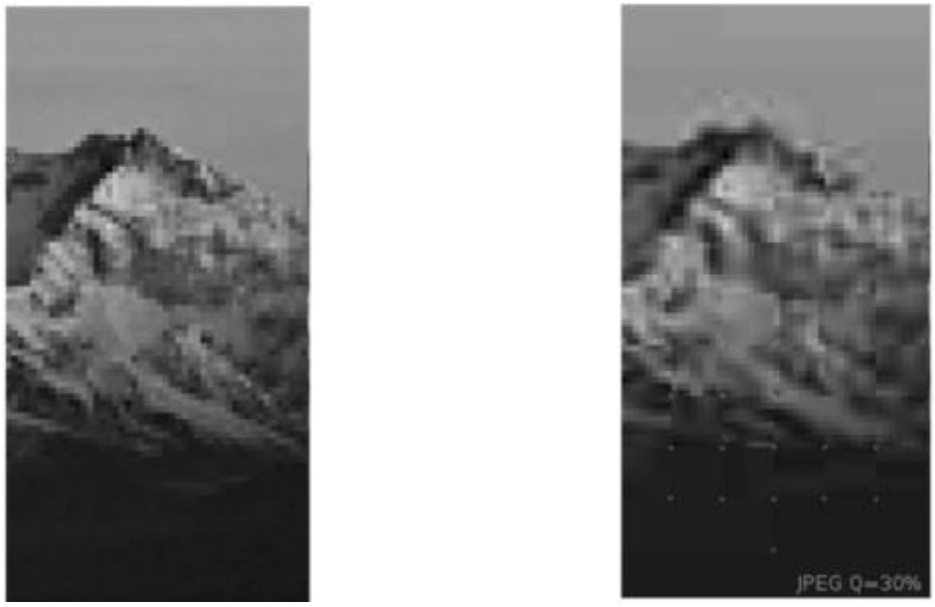


Рисунок 1.2 – Схема декодування JPEG

1.1.3 Артефакти, що виникають при стисненні алгоритмом JPEG

Оскільки в стандартному алгоритмі JPEG кожен 8x8 блок обробляється окремо, краю окремих блоків стають видні при великих коефіцієнтах стиснення. На рисунку 1.3 зображений приклад блокових артефактів. Сильна ступінь стиснення прибирає високі частоти з блоку, губляться дрібні деталі і кордони між блоками стають візуально видимими.

Алгоритм JPEG обробляє зображення в частотній області, використовуючи подання до вигляді суми ряду косинусів (ДКП). Частотна область добре підходить для представлення зображень, що мають відносно плавні переходи яскравості від однієї області до іншої, але погано підходить для подання різких перепадів яскраво 35 сти зображень. Алгоритм намагається апроксимувати різкий перепад яскравості за допомогою суми косинусів, що призводить до виникнення артефактів дзвону.



а) б)

Рисунок 1.3 – а) вихідне зображення; б) стиснуте зображення з блочними артефактами

На рисунку 1.4 представлено вихідне зображення і стислий зображення з артефактами дзвону.

а) б)

Рисунок 1.4 – а) початкове зображення; б) стиснуте зображення з артефактами дзвону

З математичної точки зору, умови появи артефактів дзвону в блоці можна описати таким чином: блок містить деяку кількість високочастотних коефіцієнтів ДКП, і цей блок розташовується поруч з блоком, велика частина ДКП коефіцієнтів якого дорівнює нулю. Поява артефактів дзвону є прояв феномена Гіббса, який полягає в проблемі чисельного представлення функції з різкими стрибками через кінцеве число базисних функцій. Проблема полягає в тому, недостатнє число використовуваних базисних функцій призводить до анулювання артефактів.

1.2 Визначення комп'ютерної обробки цифрових зображень

Комп'ютерна обробка і розпізнавання зображень є швидко розвиваючоюся самостійною дисципліною.

Комп'ютерна обробка зображень передбачає обробку цифрових зображень за допомогою комп'ютерів або спеціалізованих пристроїв, побудованих на цифрових сигнальних процесорах.

При цьому під обробкою зображень розуміється не тільки поліпшення зорового сприйняття зображень, але і класифікація об'єктів, виконувана при аналізі зображень.

У 60-ті роки минулого століття отримала розвиток особлива наука про зображення – «іконіка», яка присвячена дослідженням загальних властивостей зображень, цілей і завдань їх перетворення, обробки та відтворення, розпізнавання графічних образів.

Термін «іконка» походить від грецького «eikon», що означає зображення, образ. Сьогодні під ним розуміють «створення і обробку зображень за допомогою ЕОМ, що збігається з поняттям комп'ютерної обробки зображень.

Області застосування цифрової обробки в даний час значно розширюються, витісняючи аналогові методи обробки сигналів зображень.

Методи цифрової обробки широко застосовуються в промисловості, мистецтві, медицині, космосі. Вони застосовуються при управлінні процесами, автоматизації виявлення і супроводу об'єктів, розпізнаванні образів і в багатьох інших додатках.

Цифрова передача зображень з космічних апаратів, цифрові канали передачі сигналів зображень вимагають забезпечення передачі все більших потоків інформації. Якщо при передачі цифрового сигналу кольорового телебачення необхідно передавати потоки порядку 216 Мбіт / с, то для передачі телебачення високої чіткості швидкість передачі повинна складати близько 1 Гбіт / с. Формування зображень, поліпшення якості та автоматизація обробки медичних зображень, включаючи зображення, створювані електронними мікроскопами, рентген-апаратами, томографами і т.д., є предметом дослідження і розробки.

Сьогодні в медичній техніці широко застосовуються системи формування зображення, його перетворення в цифрову форму, візуалізація та документування шляхом введення в комп'ютер зображень за допомогою спеціалізованих пристроїв захоплення відео.

Автоматичний аналіз в системах дистанційного спостереження широко застосовується при аналізі місцевості, в лісовому господарстві, наприклад, для автоматичного підрахунку площі вирубок, в сільському господарстві для спостереження за дозріванням урожаю, при розвідці, в системах протипожежної безпеки. Контроль якості продукції, що виробляється виконується завдяки автоматичним методам аналізу сцен.

Комп'ютерна обробка зображень застосовується в задачах експертизи живопису неруйнівними методами. Для відновлення старих фільмів застосовуються методи автоматичної компенсації дефектів відеоматеріалу, отриманого після перетворення киноизображения в відео.

Сьогодні важко уявити область діяльності, в якій можна обійтися без комп'ютерної обробки зображень. Інтернет, стільниковий телефон,

відеокамера, фотоапарат, сканер, принтер, так міцно увійшли в наш побут, та всі вони немислимі без комп'ютерної обробки зображень.

При комп'ютерній обробці зображень вирішується широке коло завдань, таких як поліпшення якості зображень; вимірювання параметрів; спектральний аналіз багатовимірних сигналів; розпізнавання зображень; зменшення розміру зображень.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ПОЛІПШЕННЯ ЗОБРАЖЕННЯ І ЗАСТОСУВАННЯ ДЛЯ ЦЬОГО ШНМ

Завдання якісного поліпшення зображень є одним з найактуальніших питань цифрової обробки зображень.

Суть даного питання полягає в тому, що, маючи зображення низького роздільної здатності, необхідно отримати з нього зображення з великим розміром ширини і висоти в пікселях або з великим лінійним розміром, тобто зображення з високою роздільною здатністю.

Це завдання важливе, наприклад, для збільшення старих цифрових фотографій низького роздільної здатності. Ще одним прикладом може послужити питання поліпшення якості медичних знімків. Особливості сучасного медичного обладнання такі, що отримуване зображення може мати меншу роздільну здатність, ніж дозвіл монітора, на якому фахівець бачить знімок.

Для вирішення цього завдання застосовуються різні методи, що відносяться до класу Single image super-resolution (суперроздільна здатність на основі одного зображення). Однак найчастіше застосування методів інтерполяції, фільтрів і різних алгоритмів обробки не дає бажаного результату.

В останні роки з'являється все більше досліджень, в яких розглядається питання про використання штучних нейронних мереж для збільшення роздільної здатності.

Штучна нейронна мережа (ШНМ) – це математична модель, що описує систему з'єднаних між собою штучних нейронів, а також реалізації цієї моделі. ШНМ побудована в певному сенсі за образом і подобою біологічних нейронних мереж. ШНМ є потужним інструментом, який дозволяє досягти результатів, які можуть перевершувати використання алгоритмів і фільтрів.

При дослідженні результатів роботи під ефективністю в першу чергу розуміється якість отриманого зображення. Є безліч метрик, що дозволяють отримати об'єктивну оцінку якості поліпшеного зображення. Але, крім цього, важливо також враховувати такі фактори, як час роботи алгоритму, необхідні обчислювальні потужності, пам'ять, навантаження процесора і т.д. Однією з причин, за якої ШНМ досі не отримала широкого поширення, є повільний час обробки.

Сам по собі вибір метрик для вимірювання якості результатів, а також підхід до порівняння алгоритмів, є цікавим питанням, вартим окремого вивчення.

При появі нових моделей нейронних мереж їх часто намагаються застосувати й адаптувати до різних завдань, пов'язаних з обробкою зображень, в тому числі до задачі поліпшення зображення. Особливо це актуально для відносно нової моделі генеративнозмагальної ШНМ. Так як нові моделі з'являються постійно, а сама сфера дослідження машинного навчання і ШНМ є динамічно розвиваючоюся областю, то можна стверджувати, що ця задача є актуальною як в рамках питань цифрової обробки зображень, так і в рамках дослідження ШНМ.

2.1 Методи поліпшення зображень

Одним з основних інструментів, широко використовуваних при вирішенні завдання збільшення роздільної здатності зображень, є інтерполяція.

Інтерполяція – це спосіб знаходження проміжних значень величини за наявним дискретним набором відомих значень. Щодо збільшення зображень інтерполяції – це створення нових пікселів на основі існуючих.

Що більша роздільна здатність зображень з використанням методів інтерполяції можливе виникнення наступних дефектів (артефактів):

Ringing або Overshooting – виникнення хвильових перешкод близько різких меж зображення. Найбільш помітне на контрастних межах (наприклад, між чорним і світло-сірим кольором);

Aliasing – «сходовий ефект» – поява нерівномірності («ступінчастості») на різких діагональних межах зображення. Для усунення цього ефекту в багатьох сучасних графічних редакторах використовуються методи згладжування, що відносяться до класу Antialiasing;

Unsharpening – розмиття або втрата чіткості зображення;

Subpixel shift – субпиксельной зсув зображення. Може бути викликаний особливостями реалізації інтерполяційних алгоритмів. Візуально може бути не так помітний, як згадані вище артефакти, але, тим не менш, він впливає на формальні метрики оцінки якості.

Алгоритми інтерполяції діляться на 2 категорії: адаптивні та неадаптівні.

Адаптивні алгоритми змінюються в залежності від предмета і завдання інтерполяції. Неадаптивні методи, на відміну від попередніх, обробляють всі пікселі однаково. Адаптивні нелінійні методи не набули широкого поширення через специфічність їх застосування. Однак в ліцензованих програмах іноді застосовуються комерційні алгоритми на основі адаптивних методів. Прикладом таких програм можуть послужити Qimage, PhotoZoom Pro, Genuine Fractal, а також багато інших [4].

Як правило, адаптивні алгоритми дозволяють уникнути виникнення таких спотворень, як overshooting або aliasing, а також дозволяють зберегти різкість кордонів. Однак найчастіше можливе виникнення невластивих вихідного зображення одиночних пікселів або текстур.

Неадаптивні лінійні алгоритми включають в себе метод найближчого сусіда, білінійну і Бікубічні інтерполяцію, сплайни, функцію кардинального синуса і багато інших. Залежно від складності, алгоритми

можуть використовувати від 0 до 256 (або більше) суміжних пікселів для інтерполяції [6].

В лінійних методах для збільшення або зменшення зображення виробляється обчислення значень інтенсивності (яскравості) сусідніх пікселів зображення в залежності від обраного алгоритму.

У загальному випадку лінійні методи являють собою наступну згортку:

$$f(x) = \sum_{i=\infty}^{\infty} F(i)K(i-x), \tag{2.1}$$

де F (i) – яскравість і-ого пікселя на зображенні;

K – маска, що накладається на зображення.

У двовимірному випадку:

$$f(x,y) = \sum_{i.j=\infty}^{\infty} F(i,j)K(j-y). \tag{2.2}$$

Результат залежить від вибору ядра маски K [4].

Метод найближчого сусіда (Nearest neighbor) є базовим для більшості алгоритмів інтерполяції. Цей метод вимагає найменшого часу обробки, так як при обробці враховується тільки один піксель – найближчий до точки інтерполяції. В результаті збільшується розмір кожного пікселя.

Це простий і швидкий метод, однак результат його роботи, як правило, виглядає гірше в порівнянні з більш складними методами. Один з основних недоліків методу - пікселізація. Через те, що був збільшений розмір кожного пікселя, на зображенні відсутні плавні переходи між лініями, границями, а також псується якість градієнтів. Але незважаючи на

те, що метод рідко використовується на практиці, він доступний в більшості графічних редакторів.

Білінійна інтерполяція розглядає 4 найближчих відомих пікселя, що оточують невідомий. Як інтерпольоване значення використовується зважене усереднення цих чотирьох пікселів. В результаті зображення виходять більш гладкими, на них не так помітна пікселізація в порівнянні з результатами роботи методу найближчого сусіда.

В ході роботи алгоритму відбуваються дві лінійні інтерполяції: спочатку відбувається інтерполяція в одному напрямку, потім проводиться інтерполяція в перпендикулярному напрямку.

Ядро білінійної інтерполяції виглядає наступним чином:

$$f(x) = \begin{cases} 0, |x|>1 \\ 1-|x|, |x|<1 \end{cases}, \qquad (2.3)$$

де x – це відстань між двома точками, для яких буде здійснюватися інтерполяція [11].

Бікубічна інтерполяція є удосконаленням білінійної і розглядає масив з навколишніх 16 пікселів. Оскільки вони знаходяться на різних відстанях від невідомого пікселя, найближчі пікселі отримують при розрахунку більшу вагу.

Бікубічна інтерполяція, в порівнянні з попередніми двома методами, значно краще збільшує контрастні зображення. Інтерпольована поверхня більш гладка, в порівнянні з поверхнями, отриманими методами білінійної інтерполяції або інтерполяції методом найближчого сусіда. Також вона є оптимальною по співвідношенню часу обробки і якості результату. З цієї причини бікубічна інтерполяція є стандартною у багатьох програмах обробки зображень (включаючи Adobe Photoshop і Corel Photopaint), а також для драйверів принтерів і вбудованої інтерполяції фотокамер [6]. До

того ж, в Adobe Photoshop використовуються власні модифікації даного алгоритму - Bicubic Smoother і Bicubic Sharper.

Ядро бікубічної інтерполяції:

$$f(x) = \begin{cases} (a+2)|x|^3 + 1, |x| \le 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, 1 < |x| < 2, \\ 0, 2 \le |x| \end{cases} \quad (2.4)$$

де x – це відстань між двома точками, для яких буде здійснюватися інтерполяція;

a – коефіцієнт, значення якого зазвичай вибирається з діапазону [- 0.5, - 0.75] [11].

У деяких графічних редакторах (напр. В GIMP і Krita) для інтерполяції зображень використовується фільтр Ланцоша. В його основі лежить функція sinc (кардинального синуса):

$$f(x) = \sum_{i=-\infty}^{\infty} F(i)K(i-x), \quad (2.5)$$

де a – це позитивне ціле число (2 або 3) [11].

Цей метод дозволяє отримати високу чіткість зображення, а також зберегти різкі лінії. Однак в результаті його роботи часто на зображенні можуть з'явитися такі артефакти, як ringing. Навколо кордонів і ліній виникають вузькі контрастні області, що дозволяє зберегти різкість ліній при збереженні достатньої гладкості тональних переходів, тобто уникнути пікселізації. Крім обробки зображень цей метод часто використовується для обробки відео. Так, наприклад, він використовується для обробки відео на сайті потокового мовлення Twitch.tv.

Одним із сучасних методів інтерполяції зображень є DCCI (Directional Cubic Convolution Interpolation) [20], розроблений в 2013 році. Цей алгоритм включає в себе 3 етапи:

- ініціалізація парних точок значеннями зменшеного зображення;
- обчислення значень в непарних точках, за допомогою кубічної згортки з діагональними точками по відношенню до апроксимованої;
- обчислення значень в непарних-парних і парних-непарних точках, за допомогою кубічної згортки з вертикальними або горизонтальними точками по відношенню до апроксимованої.

2.2 Штучні нейронні мережі

Штучні нейронні мережі належать до сфери машинного навчання. Машинне навчання – це розділ науки про штучний інтелект, який вивчає алгоритми, які здатні навчатися на заданому наборі даних [2].

2.2.1 Загальна характеристика та визначення ШНМ

Алгоритми машинного навчання зазвичай прийнято ділити на 2 категорії [17]:

- алгоритми навчання з вчителем (Supervised learning) – для цих алгоритмів мається на увазі наявність пар «вхідні дані – правильна відповідь». Алгоритм в процесі роботи відновлює зв'язок між вхідними даними і очікуваними результатами. На основі вивченої інформації алгоритм повинен навчитися створювати результат, максимально схожий з правильним (очікуваним) результатом вихідної тестової вибірки;
- алгоритми навчання без вчителя (Unsupervised learning) – відноситься до класу задач обробки даних, в яких відомі тільки описи безлічі об'єктів навчальної вибірки. Алгоритм в ході роботи повинен

виявити внутрішні взаємозв'язки, закономірності і залежності, що існують між цими об'єктами. Прикладами таких алгоритмів можуть послужити завдання кластеризації, візуалізації даних, пошуку пропущених значень і т.д.

В рамках вирішення завдання поліпшення зображень, як правило, використовується навчання з учителем.

ШНМ є системою, яка перетворює інформацію подібно процесам, що відбуваються в мозку людини. Оброблювана інформація має чисельний характер, що дозволяє використовувати ШНМ, наприклад, в якості моделі об'єкта з абсолютно невідомими характеристиками. Нейронні мережі часто застосовуються в задачах розпізнавання, класифікації, аналізу і стиснення образів [2].

Одним з головних переваг нейронних мереж перед традиційними алгоритмами є можливість навчання. Технічно навчання полягає в обчисленні в ході роботи алгоритму коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними та вихідними даними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці. Більш того, мережа також може бути здатна видати правильний результат навіть на підставі неповних і частково перекручених даних

Структурною одиницею ШНМ є нейрон, який являє собою аналог біологічного нейрона. З точки зору математики штучний нейрон являє собою нелінійну функцію, обробну все вхідні сигнали і посилає результат на єдиний вихід нейрона. Ця нелінійна функція називається активаційною функцією або функцією активації.

Штучний нейрон має групу синапсів, які представляють собою односпрямовані зв'язку, з'єднану з попередніми нейронами. Вихідний зв'язок нейрона з синапсами наступних нейронів називається аксон.

Кожен синапс можна охарактеризувати за допомогою ваг W. Ця величина є аналогом електричної провідності біологічного нейрона

Для забезпечення паралельної обробки сигналів нейрони об'єднуються в шари в залежності від особливостей їх функціонування. Шари прийнято нумерувати зліва направо. Перший шар називається вхідним. Нейрони цього шару приймають надходять вхідні дані і передають їх наступному шару. Останній шар називається вихідним, тому що в якості вихідних даний він формує результати роботи нейронної мережі. Шари, розташовані між вхідним і вихідним шаром, називаються прихованими або внутрішніми. У цих шарах відбувається основна обробка даних.

У разі, якщо вихідні сигнали нейронів одного шару передаються всім нейронам наступного шару, шар, який одержує ці сигнали, називається повнозв'язним.

При навчанні ШНМ для визначення коректності отриманого результату роботи використовується функція втрат. У задачі навчання з учителем функція втрат може показувати різницю між реальними даними і даними, отриманими в результаті роботи нейронної сеї.

Серед різновидів машинного навчання варто також виділити категорію глибинного навчання (або інакше глибокого, Deep Learning). Цю категорію виділяє велику кількість прихованих шарів.

Однією з основних характеристик ШНМ є її модель. Охарактеризувати цю модель можна за видами нейронів, структурі моделі і способам навчання.

Для завдання поліпшення зображень використовуються різні моделі ШНМ, однак найкраще себе зарекомендували такі: згорткова нейронна мережа (Convolutional neural network, CNN) і генеративнозмагальна мережа (Generative adversarial network, GAN).

2.2.2 Згорткова ШНМ

Згорткові нейронні мережі – це багатошарові ШНМ прямого поширення, що складаються з великої кількості перцептронів. У загальному випадку згорткова нейронна мережа складається з ланцюжка декількох згорткових шарів, які використовуються для виділення ознак, і декількох повнозв'язних шарів. Запропонована Яном Лекуном в 1988 році, входить до складу технологій глибинного навчання.

В основі згортальних нейронних мереж лежать такі ідеї:

- кожен нейрон шару отримує вхідний сигнал від локального рецептивного поля в попередньому шарі. Під рецептивним полем розуміється шар нейронів, що сприймають зовнішні сигнали і передають їх наступного шару. При цьому рецептивне поле безпосередньо взаємодіє з наступним шаром;
- кожен прихований шар нейронної мережі складається з безлічі карт ознак (feature maps). У них всі нейрони мають загальні ваги;
- за кожним шаром згортки слідує обчислювальний шар, який здійснює локальне усереднення карт і просторову підвибірку.

При обчисленні мережі виходить, що кожен нейрон виконує операцію згортки (Конволюції) деякої області попереднього шару, яка визначається безліччю нейронів, пов'язаних з даним нейроном. Шари нейронної мережі, побудовані таким чином, називаються згортковими шарами.

Крім, згорткових шарів в згортковій нейронній мережі можуть бути шари субдискретизації і повнозв'язні шари. Шари субдискретизації виконують функції зменшення розмірності простору карт ознак. Даний шар дозволяє прискорити подальші обчислення. Субдискретизація можлива завдяки тому, що в цій архітектурі нейронних мереж важливо не стільки значення ознаки, скільки сам факт його наявності.

Вихідний шар згорткової нейронної мережі, як правило, завжди повнозв'язний. Нейрони цього шару мають повне з'єднання з усіма активаціями попереднього шару.

Всі три види шарів можуть чергуватися в довільному порядку. Це дозволяє складати карти ознак з карт ознак, що на практиці означає здатність розпізнавання складних ієрархій ознак. Це дозволяє поширитися від конкретних особливостей вхідних даних до абстрактних деталей.

При навчанні нейронних мереж зазвичай використовується метод зворотного поширення помилки (Backpropagation) або його модифікації. Суть даного методу полягає в тому, що поширення сигналів помилки в ШНМ походить від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів.

Даний вид нейронних мереж дуже добре зарекомендував себе для вирішення безлічі завдань обробки зображень [3].

Однією з найбільш відомих моделей згорткових мереж, що застосовуються для вирішення завдання поліпшення зображень, є розроблена в 2014 році модель нейронної мережі SRCNN (Super-Resolution Convolutional Neural Network) [14].

На вхід SRCNN подається зображення, збільшене за допомогою методів інтерполяції, яке потім проходить процес поліпшення якості. Подібний метод застосовується в розробленому компанією Google методі RAISR [27].

Існують безліч модифікацій даної моделі, одними з найшвидших за швидкістю і ефективних за якістю є VDSR (Very Deep Convolutional Network) [14] і FSRCNN (Fast Super-Resolution Convolutional Neural Network) [9].

Також заслуговує на увагу модель ESPCN (Efficient Sub-Pixel Convolutional Neural Network) [18], яка здатна в реальному часі збільшувати зображення. На відміну від SRCNN, на вхід подається не збільшене за допомогою інтерполяції зображення, а зображення низького

дозвіл. Збільшення розмірів зображення відбувається в результаті роботи нейронної мережі.

2.2.3 Генеративнозмагальні ШНМ

Генеративнозмагальні мережі – це алгоритм машинного навчання без учителя, побудований на комбінації з двох нейронних мереж, як правило згорткових або прямого поширення. Перша мережа (генератор) генерує зразки. Друга мережа (дискримінатор) приймає на вхід результати роботи першої мережі і правильні («справжні») дані, а потім намагається відрізнити справжні зразки від згенерованих. Вперше ця модель була описана співробітниками компанії Google в 2014 році [12].

Складність навчання даної моделі полягає в тому, що, по-перше, необхідно вивчити відразу дві нейронних мережі, а по-друге, необхідно правильно налаштувати баланс між ними. Генератор навчає створювати більш якісний результат, дискримінатор навчає розпізнавати реальні дані від згенерованих. Якщо одна з мереж буде навчена краще, ніж інша, то модель в цілому не буде працювати коректно.

Модель навчається методом зворотного поширення помилки. Для навчання зазвичай використовується наступний алгоритм:

- проводиться підготовка реальних даних. У задачі збільшення зображень – це вихідні зображення, до яких не було застосовано масштабування;
- генерується шум, на базі якого генератор генерує дані;
- формується набір даних для навчання дискримінатора. Цей набір який складається зі справжніх даних, підготовлених на кроці 1 (їм присвоюється мітка 1) і підробок від генератора, отриманих на попередньому кроці (їм присвоюється мітка 0);
- проводиться навчання дискримінатора, в процесі якого на вхід подаються реальні дані і створені генератором;

- проводиться навчання генератора. На даному етапі навчається генератор, а навчання дискримінатора відключено. На вхід подається шум, на виході очікується отримання позначки 1, т. Е. Розпізнавання дискримінатором даних як реальних. При навчанні генератора не використовуються реальні зображення, а використовується тільки відмітка дискримінатора.

На даний момент однією з останніх моделей, розроблених для вирішення завдання збільшення роздільної здатності зображення за допомогою ШНМ, є SRGAN (Super-Resolution Generative Adversarial Network) [16].

2.3 Опис моделі SRCNN

Перед тим, як здійснювати поліпшення зображення за допомогою даної моделі, зображення попередньо збільшується до необхідних розмірів. Як правило, при цьому використовується алгоритм бікубічної інтерполяції. Хоча формально цей метод інтерполяції вдає із себе згортку, а самі автори дослідження [10] стверджують, що до нього може бути застосоване формулювання «шар», в описі архітектури нейронної мережі ця операція зазвичай не фігурується.

Архітектура SRCNN представлена на рисунку 2.1.

Позначимо зображення, отримане в ході інтерполяції, буквою Y, а 23 зображення високої роздільної здатності, з яким буде здійснюватися порівняння – X. Зображення Y, незважаючи на відповідність розмірам зображення X, будемо вважати зображенням низької роздільної здатності. На рисунку 2.1 зображення Y позначено написом «Low resolution image (input)».

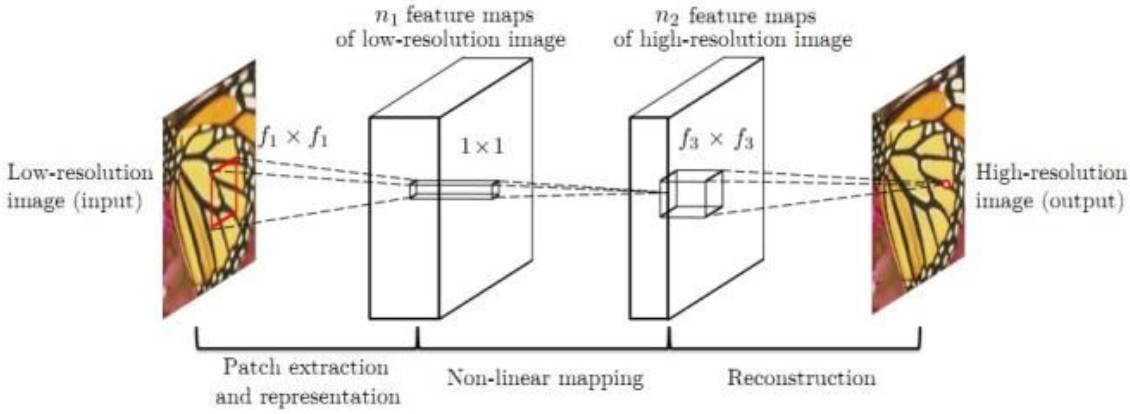


Рисунок 2.1 – Архітектура SRCNN

В ході роботи нейронної мережі необхідно отримати зображення F(Y) настільки схоже на зображення X, наскільки це можливо. Для цього необхідно знайти функцію F. Пошук складається з наступних 3-х операцій:

- отримання патчів (рис. 2.1) позначено написом «Patch extraction and representation». Ця операція дозволяє отримати перекриваючі патчі (окремі ділянки зображення, що складаються з декількох пікселів) із зображення низької роздільної здатності Y, а потім представити кожен патч у вигляді багатовимірного вектора. Ці вектори містять набір карт ознак, розмір яких дорівнює розмірності вектора.
- нелінійне відображення (рис. 2.1) - це операція позначена написом «Non-linear mapping». Результатом цієї операції є нелінійне відображення кожного багатовимірного вектора на інші вектори. Кожен відображений вектор являє патч з високою роздільною здатністю. Ці вектори містять інший набір карт ознак.
- реконструкція (рис. 2.1) позначена написом «Reconstruction». Під час виконання цієї операції відображення з високою роздільною здатністю кожного патча, отримані в попередньому кроці, використовуються для того, щоб згенерувати зображення високої роздільної здатності F (Y). Очікується, що отримане зображення буде схожим з вихідним зображенням X. Отримане в результаті роботи

нейронної мережі зображення F (Y) позначено (рис. 2.1) написом «High-resolution image (output)».

Всього дана модель містить три шари.

Для отримання патчів використовується перший шар нейронної мережі, який можна позначити формулою:

$$F_1(Y) = \max(0, W_1 * Y + B_1), \tag{2.6}$$

де W1 – фільтри;

B1 – ваги;

* – операція згортки.

Фільтри W1 можна позначити через формулу:

$$W_1 = c \times f_1 \times f_1, \tag{2.7}$$

де c – кількість каналів зображення Y;

f1 – просторовий розмір фільтра.

Фільтри W1 здійснюють n1 операцій згортки зображення, і кожна згортка має ядро згортки розміру c x f1 x f1. Вихідні дані шару містять n1 карт ознак.

Ваги B1 є n1-мірний вектор, кожен елемент якого зіставлений з елементом фільтра W1. Як активаційні функції першого шару використовується ReLU (Rectified Linear Unit, «випрямляч»):

$$f(x) = \max(0, x), \tag{2.8}$$

де x – вхідний сигнал.

Другий шар здійснює операцію нелінійного відображення n1- мірний векторів на n2-мірні вектори. Ця операція працює тільки для фільтрів з розміром 1 х 1.

Операцію, здійснювану другим шаром, можна висловити через наступну формулу:

$$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2), \tag{2.9}$$

де параметри W2 і B2 аналогічні схожим параметрам для F1 з тією різницею, що тепер фільтри здійснюють n2 операцій, а ваги є n2-мірний вектор.

Як активаційні функції, також як і у першого шару, використовується ReLU.

Третій шар здійснює операцію реконструкції. Його можна представити наступною формулою:

$$F(Y) = W_3 * F_2(Y) + B_3, \tag{2.10}$$

де W3 – це c фільтрів розмірністю n2 х f3 х f3;

B3 – це c-мірний вектор.

Вихідними даними цього шару є зображення з високою роздільною здатністю.

Функцією активації даного шару служить лінійна функція:

$$f(x) = x \tag{2.11}$$

де х – це вхідний сигнал.

В якості опції втрат використовується середньоквадратична помилка MSE (Mean Squared Error):

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left|F(Y_i) - X_i\right|^2 \tag{2.12}$$

де θ – це параметри нейронної мережі {W1, W2, W3, B1, B2, B3};

n – кількість зразків в навчальній вибірці.

Авторами даної моделі були проведені тестування для різних значень показників. В якості найбільш оптимальних були виявлені наступні значення: f1 = 9, f2 = 1, f3 = 5, n1 = 64 і n2 = 32 [10].

2.4 Опис моделі SRGAN

Вхідними даними служить зображення. Позначимо зображення, яке ми отримуємо в результаті роботи нейронної мережі, як I SR, зображення низької роздільної здатності, яке необхідно збільшити в n раз, як I LR. Зображення з високою роздільною здатністю (без обробки) позначимо як I HR. I LR виходить в результаті зменшення I HR в n раз.

В результаті вивчення необхідно отримати генеруючу функцію G. Генератор навчається як згорткова нейронна мережа прямого поширення з параметрами ΘG:

$$\theta_G = \left\{W_{1:L}; b_{1:L}\right\}, \tag{2.13}$$

де $W_{1:L}$ і $b_{1:L}$ – це, відповідно, ваги і зміщення нейронної мережі глибиною в L шарів. Цей параметр обчислюється шляхом оптимізації складовою функції втрат lSR. Для вивчення нейронної мережі знаходимо значення:

$$\theta_G = \min \frac{1}{N} l_{SR}(G(I_n^{LR}), I_n^{HR}, \quad (2.14)$$

Функція втрат lSR складається з наступних компонентів:

$$l_{SR} = l_{MSE} + 10^{-3} l_{GEN}, \quad (2.15)$$

де l_{MSE} – середня квадратична помилка при порівнянні згенерованого зображення I SR з його оригіналом I HR;

l_{GEN} – змагальна функція втрат, яку можна представити наступною формулою:

$$l_{Gen} = \sum_{n=1}^{N} -\log(D(G(I^{LR}))), \quad (2.16)$$

де D (G (I^{LR})) – це ймовірність розпізнавання дискримінатором того, що створене генератором зображення G (I^{LR}) є зображенням високої роздільної здатності I^{HR}.

Варто зазначити, що автори дослідження пропонують використовувати при порівнянні зображення не MSE, а попередньонавчену нейронну мережу VGG-19, тому що результати її роботи більше відповідають тому, як людське око сприймає різницю між двома зображеннями [16].

Автори даної моделі застосовують нормалізацію до вхідних даних нейронної мережі, тому що це дозволяє збільшити ефективність роботи алгоритму. Детально про переваги нормалізації розказано в статті [13].

При навчанні використовується оптимізатор Adam (Aadaptive Moment Estimation). Цей метод докладно описаний в статті [8].

Архітектура генеративної нейронної мережі (G) складається з наступних блоків: 2 згорткових шари з ядром 3x3, 64 карти ознак, за якими слідують шари нормалізації і два субпіксельних згорткових шари. Як функції активації використовується Parametric ReLU:

$$f(x) = \begin{cases} x, x>0 \\ ax, x\leq 0 \end{cases}, \tag{2.17}$$

де x – вхідний сигнал,

a – коефіцієнт, який обчислюється в результаті навчання моделі.

Для прискорення роботи нейронної мережі використовується метод батчнормалізації (Batch Normalization), запропонований співробітниками компанії Google в 2015 році [13]. Іноді зустрічається переклад цього терміна «пакетна нормалізація». Цей метод передбачає додавання додаткового шару, який дозволяє знизити спотворення даних шляхом їх нормалізації. Етоп метод дозволяє скоротити час навчання за рахунок використання більш високого значення параметра швидкості навчання (learning rate), а також знизити перенавчання.

Архітектура генератора представлена на рисунку 2.2.

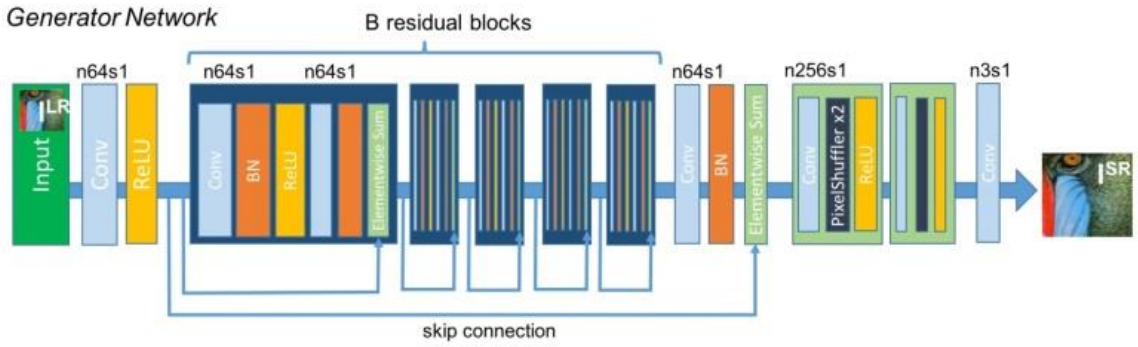


Рисунок 2.2 – Архітектура генератора

На рисунку 2.2 зображені наступні елементи:

- Input I LR – зображення низької роздільної здатності, яке подається на вхід генератора;

- Conv – згортковий шар з n картами ознак (feature maps) і кроком (stride), рівним s;

- ReLU – активаційна функція Parametric ReLU;

- BN – Batch Normalization, Батч-нормалізація;

- skip connection – додаткові зв'язку;

- B residual blocks – залишкові блоки;

- Elementwise sum – поелементно сума;

- PixelShuffler x2 – субпиксельной згорткові шари;

- I_{SR} – зображення високої роздільної здатності, отримане в результаті роботи нейронної мережі.

Щоб відрізнити зображення з високою роздільною здатністю I_{HR} від згенерованих зображень I_{SR}, здійснюється навчання дискримінатора (D).

На вхід дискримінатора в випадковому порядку подаються або зображення I_{HR} і I_{SR}. В результаті отримуємо 0 в разі, якщо зображення було розпізнано як I_{SR} і 1 в разі, якщо розпізнано як I_{HR}.

Мережа складається з 8 згорткових шарів з ядром 3x3, кількість яких збільшується в 2 рази з 64 до 512, за якими слідують 2 повнозв'язних шари. Як активаційні функції згорткових шарів використовується Leaky ReLU з a= 0,02:

$$f(x) = \frac{1}{1+e^{-x}}, \tag{2.18}$$

де x – вхідний сигнал.

Активаційною функцією для другого повнозв'язного шару служить сигмоїда:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.19}$$

де x – вхідний сигнал.

Також, як і в генераторі, в дискримінаторі використовується батчнормалізація.

Архітектура дискримінатора зображена на рисунку 2.3.

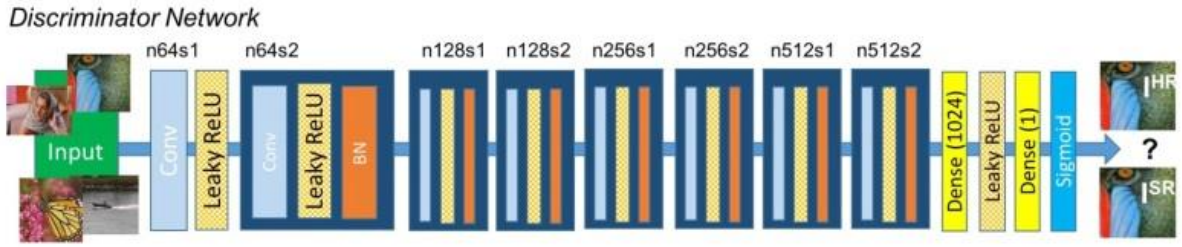


Рисунок 2.3 – Архітектура дискримінатора

На рисунку 2.3 зображені наступні елементи:

- Input – вхідні дані, зображення з високою роздільною здатністю і зображення, створені генератором;

- Conv – згортковий шар з n картами ознак (feature maps) і кроком (stride), рівним s;

- Leaky ReLU – активаційна функція Leaky ReLU;

- BN – Batch Normalization, Батчнормалізація;

- Dense – повнозв'язний шар;

- Sigmoid – активаційна функція сигмоїда.

I_{HR}, I_{SR} – зображення високої професійності і зображення низької роздільної здатності відповідно; певний дискримінатором клас зображення – справжнє зображення з високою роздільною здатністю або створене генератором.

2.5 Вибір метрик для порівняння результатів роботи алгоритмів і ШНМ

Для коректного порівняння якості роботи алгоритмів і нейронних мереж необхідно використовувати об'єктивні метрики. Вхідними даними для кожного методу буде служити зменшене зображення. Отримане в результаті роботи методів зображення можна порівняти з оригіналом на предмет появи розбіжностей. Тобто чим більше результат схожий на оригінал, тим вище якість роботи алгоритму.

Одними з найбільш поширених методів порівняння двох зображень є:

- метод середньоквадратичної помилки моделі (Mean Squared Error, MSE);
- пікове відношення сигналу до шуму (Peak Signal-to-Noise Ratio, PSNR);
- індекс структурної подібності (Structural Similarity, SSIM).

Найпростішим методом є MSE. В його основі лежить попіксельне порівняння двох зображень. Загальна формула:

$$MSE(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 , \qquad (2.20)$$

де x – і-й піксель зображення x;

y – і-й піксель зображення y;

n – загальна кількість пікселів.

При порівнянні двох зображень значення MSE, рівне нулю, говорить про те, що зображення повністю ідентичні. Чим вище значення цього показника, тим більше відмінностей між зображеннями

Як правило, метрики розрахована на вікна розміром 8 × 8 пікселів. SSIM визначається локально, тому, щоб отримати глобальну оцінку подібності двох зображень використовується усереднення SSIM по всіх вікон.

Результат, який виходить за допомогою методу SSIM, лежить в діапазоні від -1 до 1. Значення, рівне 1, досягається тільки при повній ідентичності зображень. Чим ближче значення до -1, тим більше відмінностей між зображеннями.

Загальна формула має такий вигляд:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \quad (2.21)$$

де μx – середнє зображення x;

μy – середнє зображення y;

σ2x – дисперсія зображення x;

σ2y – дисперсія зображення y;

σxy – коваріація x і y.

C1 і C2 - поправочні коефіцієнти:

$$c_1 = (k_1, 1)^2$$
$$c_2 = (k_2, 1)^2 \quad (2.22)$$

де L – динамічний діапазон пікселів

Пікове співвідношення сигналу до шуму (PSNR) показує співвідношення між максимумом можливого значення сигналу і потужністю шуму, спотворює значення сигналу. Так як багато сигналів мають широкий динамічний діапазон, PSNR прийнято зазвичай вимірювати за логарифмічною шкалою в децибелах. PSNR може приймати

значення від нуля до ∞. Чим більше значення PSNR, тим більшу схожість мають порівнювані зображення.

Можна визначити через середньоквадратичну помилку (MSE). Формула має такий вигляд:

$$PSNR(x, y) = 10 \lg \frac{L^2}{MSE(x, y)}, \tag{2.23}$$

де x,y – зображення, які необхідно порівняти,

MSE(x,y) – среднеквадратическая помилка двох зображень,

L – максимальне значення яскравості.

Зазвичай дорівнює 255 при використанні цілих чисел від 0 до 255, або 1, якщо використовуються речові значення від 0 до 1 [19].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір мови програмування і бібліотек

Для вирішення різних завдань машинного навчання і обробки зображень використовуються різні мови програмування. Також існує велика кількість різних бібліотек, в яких реалізовані найбільш часто використовувані методи і алгоритми. В якості мови програмування мною був обраний Python. На цей вибір вплинуло безліч факторів, серед яких можна виділити наступні:

- ця мова є однією з найпопулярніших мов програмування в академічному середовищі при використанні завдань, пов'язаних з машинним навчанням і інтелектуальною обробкою даних [29]. Таким чином, програмна реалізація з використанням цієї мови буде зрозуміла і доступна широкому колу дослідників;
- для даної мови програмування існує велика кількість бібліотек, які спрощують роботу зі складними алгоритмами і методами машинного навчання. Також для Python існує безліч бібліотек для роботи з зображеннями. Варто також відзначити легкість установки бібліотек за допомогою вбудованих в мову менеджерів пакетів;

Python є мовою програмування високого рівня, яка дозволяє швидко проектувати прототипи програм, що дозволяє скоротити час розробки;

Python підтримує модульність за рахунок того, що кожен файл в проекті може бути виконаний як окремий скрипт, так і бути використаним при імпорті окремих методів.

Однак у Python є і мінуси. Через те, що вона є інтерпретованою мовою програмування, алгоритми, реалізовані нею, як правило, працюють повільніше аналогічних алгоритмів, реалізованих на компільованих мовах програмування. Але цей мінус частково компенсується використанням

бібліотек, в яких була проведена оптимізація з використанням коду, на інших мовах.

При написанні програмної реалізації використовувалася версія мови 3.6.2. Це була найактуальніша стабільна версія мови на момент початку написання програми.

Для програмної реалізації було вирішено використовувати такі основні бібліотеки:

- NumPy;
- OpenCV;
- TensorFlow.

Також використовувалися різні допоміжні пакети, такі, як tqdm для візуалізації прогресу в командному рядку, shutil для роботи з файлами і папками і datetime для визначення часу роботи алгоритмів.

NumPy – це бібліотека для мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. Крім роботи з масивами, в даній бібліотеці реалізовано безліч таких корисних речей, як функції лінійної алгебри, перетворення Фур'є, генерації випадкових чисел і т.д. [25].

Як уже згадувалося раніше в цій главі, Python, будучи інтерпретованою мовою програмування, як правило, обробляє складні математичні алгоритми повільніше, ніж компільовані мови. Автори NumPy постаралися вирішити цю проблему, запропонувавши модуль з виконуваним кодом функцій на мовах C і Fortran.

Бібліотека NumPy настільки популярна, що найчастіше інтегрована в інші бібліотеки. Так, наприклад, функції з бібліотеки OpenCV можуть обробляти масиви з бібліотеки NumPy.

При створенні програмної реалізації використовувалася версія NumPy 1.13.3.

OpenCV – це бібліотека з відкритим вихідним кодом, що містить набір типів даних, функцій і класів для обробки зображень алгоритмами комп'ютерного зору. Написана і оптимізована на C / C ++, також має інтерфейси для Python, Java, Ruby, Lua та інших мов [26].

Ця бібліотека являє собою набір модулів. Кожен з цих модулів пов'язаний з певною областю комп'ютерного зору.

Алгоритми, реалізовані в цій бібліотеці, протестовані і оптимізовані, а значить, подібно бібліотеці NumPy, дозволяють частково зменшити втрати продуктивності, пов'язані з використанням інтерпретатора Python. Бібліотека включає в себе більше 1000 функцій і алгоритмів.

OpenCV є досить широко використовуваною бібліотекою завдяки її вільної ліцензії BSD. Вона застосовується такими компаніями, як NVidia, WillowGarage, Intel, Google. Компанії NVidia і WillowGarage частково спонсорують її розробку. Серед проектів, в яких використовується OpenCV, можна відзначити аудіо-візуальну інсталяцію в Музеї сучасного мистецтва в Сан – Франциско, проект з контролю якості монет, виготовлених Центробанком Китаю, панорами вулиць в проекті Google Maps [7].

У програмній реалізації була використана версія бібліотеки 3.3.0.

Так як OpenCV, в основному, реалізує тільки базові операції з зображеннями, її можна розглядати як низькорівневу бібліотеку. Для вирішення завдань машинного навчання при обробці зображень її, як правило, недостатньо. Необхідно використовувати допоміжні бібліотеки.

TensorFlow – бібліотека для глибокого машинного навчання, що розробляється в Google Brain. До 2015 року розроблялася в закритому режимі під назвою DistBelief. 9 листопада 2015 була відкрита для вільного доступу.

Робота с TensorFlow заснована на побудові та виконанні графа обчислень. Граф обчислень – це конструкція, яка описує те, яким чином будуть проводитися обчислення. На відміну від класичного написання

коду, який буде виконуватися через підрядник, в TensorFlow код необхідний для створення структури, яка задає порядок обчислень. Таким чином, в структурі програми лежать дві основні частини: складання графа обчислень і виконання обчислень в створених структурах.

Граф обчислень складається з тензорів, які вдають із себе багатовимірні масиви. Елементами тензорів служать плейсхолдери, змінні та постійні операції. Тензор може представляти із себе як окреме число, вектор ознак з розв'язуваної задачі або зображення, так і цілий набір описів об'єктів або масив із зображень. Замість одного об'єкта можливо передати в граф масив об'єктів і для нього буде вирахувано масив відповідей. Обчислювальні графи виконуються в сесіях. Об'єкт сесії приховує в собі контекст виконання графа – необхідні ресурси, допоміжні класи, адресні простори [21].

Також великим плюсом даної бібліотеки є те, що в ній є вбудований об'єкт-серіалізатор, що дозволяє зберігати і читати з файлу поточний стан графа.

У програмній реалізації використана версія Tensorflow 1.3.0.

Структуру програми складають незалежні модулі (папки), які в свою чергу містять один або кілька скриптів. Всі модулі звертаються до одного джерела даних, розташування якого, разом з іншими настройками і глобальними змінними, зберігається в окремому файлі конфігурації.

Програма була написана для операційної системи Windows 10, проте завдяки платформ використаних інструментів, вона може бути запущена на Linux-подібних операційних системах.

3.2 Підготовка бази зображень

При навчанні штучних нейронних мереж для отримання хорошого результату важливо підібрати досить велику кількість якісного матеріалу

для навчання і тестування. Для того, щоб алгоритм коректно справив обробку, зображення повинні бути одного формату, якості та розміру.

Існує велика кількість вільних ресурсів з базами зображень, таких, як, наприклад, CIFAR-10, MNIST, Labelme і т.д. Більшість цих баз поширюються в вигляді .gz або .zip архівів.

Для використання в проекті мною була обрана база зображень ImageNet [23]. Це база, яка складається більш ніж з 14 мільйонів фотографій, проаналізованих людьми. Кожній фотографії присвоєна мітка, що позначає наявність на фотографії певної ознаки або об'єкта.

Вибір цієї бази зображень пояснюється наступними причинами:

- в цій базі поширюються не зображення, а посилання на них. Це дає можливість попередньої обробки зображень на етапі їх збереження. Наприклад, можна обмежити кількість завантажуваних зображень потрібною кількістю або, наприклад, при збереженні зменшувати розміри зображень. Таким чином можливо заощадити місце на жорсткому диску, що може бути істотним плюсом при роботі з великою кількістю зображень;
- є можливість вказати тематику зображень. У цій базі представлені такі категорії, як фотографії людей, тварин, медичні зображення, фотографії природи і т.д. Щоб змінити тематику навчальної вибірки, досить замінити файл з посиланнями.

Для тестування були обрані кольорові фотографії тварин, тому що такі зображення містять велику кількість деталей.

Зображення з даної бази було вирішено зберігати в форматі JPEG з колірним простором RGB, тому що цей формат є одним з найбільш підходящих для зберігання реалістичних зображень з великою кількістю деталей. Зображення при цьому не займають багато пам'яті на диску. Для прискорення часу навчання і тестування було прийнято використовувати відносно невеликий розмір зображень – 96 x 96 пікселів.

3.3 Опис програмної реалізації

3.3.1 Загальна структура програми

Програма складається з наступних модулів:

- модуль підготовки зображень;
- модуль алгоритмів інтерполяції;
- модуль генеративної ШНМ SRGAN;
- модуль згорткової ШНМ SRCNN;
- модуль порівняння зображень.

Кожен модуль є незалежним від інших і знаходиться в окремій папці.

Рішення зробити програму з декількох модулів було прийнято після вивчення існуючих реалізацій різних моделей. Вони часто були написані таким чином, що навіть невеликі зміни параметрів вимагали серйозного рефакторінга коду. До того ж, модульність програми і централізоване зберігання всіх параметрів дозволяє з мінімальними змінами підключати до проекту нові моделі ШНМ і алгоритми інтерполяції. Так, наприклад, модуль інтерполяції і модуль згорткової нейронної мережі були створені на основі вже існуючих реалізацій і підключені до проекту в дуже короткі терміни. Таким чином вдалося частково автоматизувати процес порівняння і оцінки роботи різних алгоритмів.

У кореневому каталозі програми знаходиться конфігураційний файл config.py. Зміст даного файлу наведено в додатку А. У цьому файлі вказані загальні для всіх модулів налаштування, а також налаштування, що вказують джерело даних. У ньому перераховані шляхи до папки з вихідними зображеннями, розмір зображень, розмір Батче (пакета вихідних даних) при навчанні і т.д.

Всі модулі використовують одне й теж джерело даних, яке розташоване в папці data. У ній в папці train знаходяться вихідні зображення з високою роздільною здатністю, які використовуються при навчанні нейронних мереж, в папці test – зображення високої роздільної

здатності, які використовуються при тестуванні. У папках train_low і test_low знаходяться зображення низької роздільної здатності, які використовуються в якості вхідних даних для алгоритмів збільшення зображень і моделей ШНМ при навчанні і тестуванні відповідно.

Раніше було прийнято використовувати готову реалізацію згорткових ШНМ. Так як в кінцевому підсумку ця реалізація була додана в проект з мінімальними змінами, в цій главі не буде наводиться опис модуля SRCNN. Код даної моделі можна скачати зі сховищ автора [26].

Інтерфейс взаємодії з користувачем реалізований за допомогою простої консолі, в якій відображається різна службова інформація і прогрес роботи алгоритмів. Результати роботи окремих модулів зберігаються в окремих файлах.

Кожен модуль програми незалежний від інших і може бути запущений окремо.

3.3.2 Модуль підготовки зображень

Модуль prepare_data відповідає за підготовку зображень. У нього входять 2 скрипта: download_images.py і prepare_data.py. Скрипт download_images.py відповідає за скачування, збереження і первинну обробку зображень. Код даного скрипта можна подивитися в додатку Б.

В папку з цим скриптом необхідно помістити файл links.txt з посиланнями на зображення з бази ImageNet. Файл з посиланнями можна скачати з сайту цієї бази. Можна додати посилання на зображення з інших баз, однак важливо враховувати, що кожне нове посилання повинне починатися з нового рядка і при цьому в тексті не повинно бути ніяких зайвих елементів (нумераций, назв зображень і т.д.) Для того, щоб використовувати інший файл з посиланнями, необхідно замінити значення змінної links_file в файлі config.py.

В ході реалізації виникла наступна проблема: деякі зображення були не доступні по посиланнях, вказаним в базі. Такі випадки можна було визначити по одній з таких ознак:

- змінилося посилання на зображення, тобто сталася переадресація на сторінку, що повідомляє про те, що зображення більш не доступно. Проблема була вирішена порівнянням посилання, що міститься у файлі links.txt, з посиланням, з якої фактично було проведено завантаження зображення;
- зображення було збережено з розмірами 1x1 піксель. Причина виникнення даної проблеми - помилка на сервері із зображенням. Проблема була вирішена перевіркою розмірів зображення по висоті і ширині з розмірами, заданими в файлі конфігурації. Також це дозволило уникнути появи у вибірці зображень із занадто маленькою розподільною здатністю.

Таким чином, скрипт download_images.py перевіряє кожне посилання з файлу links.txt і перевіряє, щоб ширина і висота зображення не була менших розмірів, ніж зазначено в файлі конфігурації. Для зміни розмірів зображень та їх збереження в формат jpg використовуються функції бібліотеки OpenCV.

Первинна обробка зображень включає в себе зменшення зображення і операцію кадрування до розмірів, заданих в файлі конфігурації. Зменшення розмірів проводиться відповідно до масштабу, зазначеного в файлі конфігурації змінною scale_factor. Щоб зменшити роздільну здатність був використаний алгоритм бікубічної інтерполяції.

Скрипт prepare_data.py здійснює збереження завантажених зображень в форматі масиву numpy в файл з роздільною здатністю npy. Це необхідно для того, щоб при навчанні не вироблялося повторне зчитування і обробка всіх зображень, а відразу отримувалася інформація у вигляді багатовимірних масивів. За рахунок цього вдається трохи скоротити час

роботи нейронної мережі. Вихідний код цього скрипта можна подивитися в додатку В.

В даному модулі використані бібліотеки OpenCV (для збереження зображень), NumPy для перетворення скачаних зображень в масив, з яким може працювати OpenCV і бібліотека shutil для роботи з папками. Для скачування зображень використовується стандартна бібліотека Python urllib.request.

3.3.3 Модуль алгоритмів інтерполяції

Цей модуль містить в собі реалізацію збільшення зображень за допомогою алгоритмів бікубічної інтерполяції і фільтра Ланцоша. Для цього використовуються функція бібліотеки OpenCV resize з параметрами interpolation = cv2.INTER_CUBIC для бікубічної інтерполяції і interpolation = cv2.INTER_LANCZOS4 для фільтра Ланцоша. Модуль містить єдиний скрипт interpolation.py, текст якого можна подивитися в додатку Г. Результати роботи кожного алгоритму зберігаються в окремій папці, шлях до якої задається в конфігураційному файлі.

3.3.4 Модуль порівняння зображень

Даний модуль складається з одного скрипта compare.py, в якому реалізовані метрики порівняння зображень PSNR, SSIM і MSE. Зміст скрипта представлено в додатку Д.

Незважаючи на те, що існують бібліотеки, в яких вже є готові реалізації цих метрик, мені не вдалося знайти бібліотеку, в якій були присутні готові рішення одночасно для всіх 3. З цієї причини, а також щоб уникнути створення нових залежностей від сторонніх бібліотек, було прийнято рішення написати власну реалізацію цих метрик.

У цьому модулі були реалізовані функції getMSE, getPSNR і getSSIM, в які передаються масиви, отримані за допомогою функції бібліотеки OpenCV cv2.imread. Ці функції повертають значення відповідної метрики.

Модуль порівнює оригінальні зображеннями, що зберігаються в data / test з зображеннями, отриманими в результаті роботи алгоритмів і ШНМ. Шлях до папок з згенерованими зображеннями вказується в файлі конфігурації.

Отримані результати зберігаються у вигляді csv-файлів. Цей формат дозволяє здійснювати швидке завантаження результатів в різні табличні редактори для їх подальшого аналізу та подання. Також крім цього відбувається друк в вікно консолі середніх результатів по всіх метрик для кожного алгоритму.

У цьому модулі використані бібліотеки OpenCV і NumPy для проведення математичних операцій над зображеннями. OpenCV також використовується для читання зображень. Крім цього, був використаний стандартний пакет Python csv для збереження результатів порівняння зображень у вигляді csv-файлу.

3.3.5 Модуль генеративнозмагальної ШНМ

Цей модуль містить скрипти і код, необхідний для роботи реалізованої моделі ШНМ SRGAN.

Модуль складається з наступних файлів: srgan.py, test.py і train.py. Крім цих файлів модуль містить папку results, в яку відбувається збереження отриманих в результаті роботи моделі зображень та папку backup, в якій зберігається проміжний або фінальний стан моделі в залежності від того, чи був закінчений процес навчання чи ні.

У всіх файлах модуля використовується бібліотека TensorFlow.

Крім цього, для зручності читання коду були написані методи обертки. Вивчення різних існуючих реалізацій показало, що виділення таких функцій в окремий модуль або клас є досить поширеною практикою. До того ж існують бібліотеки, які представляють собою надбудови над існуючими функціями з TensorFlow, наприклад, пакет TensorLayer. Ці функції винесені в файл layer.py. При написанні цих функцій була частково використана наступна реалізація цієї моделі [28].

У файлі srgan.py реалізована модель SRGAN у вигляді окремого класу. Діскрімінатор і генератор реалізовані у вигляді методів класу discriminator і generator.

У конструктор моделі передаються такі параметри: data, low_res, is_training, batch_size.

Параметр data – це масив numpy.ndarray, що представляє набір тренувальних або тестових зображень в залежності від того, чи знаходиться мережа в стані навчання чи ні.

Параметр low_res – це масив зображень низької роздільної здатності, які необхідно буде збільшувати.

Параметр is_training є булевим показником стану нейронної мережі, значення True відповідає тому, що мережа знаходиться в стані навчання.

Параметр batch_size – розмір Батче (пакета), тобто кількість зображень, які буде оброблено мережею за одну ітерацію.

В якості вхідних даних в метод generator передаються параметри data і is_training. У метод discriminator передаються параметри data, is_training, reuse. Параметри data і is_training аналогічні параметрам, переданим в конструктор.

Параметр reuse дозволяє не створювати нові змінні, а використовувати вже згенеровані. Структура і порядок шарів в генераторі і дискримінаторі повністю відповідають моделі, яка описана в [16].

У файлі train.py реалізована функція, яка виробляє навчання нейронної мережі. Вона здійснює завантаження тестового набору даний і

даних низької роздільної здатності, проводить їх нормалізацію, а потім передає в конструктор моделі SRGAN. У цьому ж файлі вказується оптимізатор для втрат. Оскільки автори оригінального дослідження використовували оптимізатор Adam з параметром β = 1e-3, в цій реалізації використовується функція бібліотеки TensorFlow tf.train.AdamOptimizer з параметром learning_rate = 1e-3.

При запуску цього скрипта також відбувається перевірка, чи існує файл з попереднім станом мережі. Після кожної обробки всіх тестових даний відбувається збереження стану мережі. Обидві ці операції здійснюються за допомогою об'єкта бібліотеки TensorFlow tf.train.Saver. Це дозволяє переривати навчання, а потім продовжувати його з моменту, на якому довелося зупинитися в попередній раз.

Вихідний код даного файлу представлений в додатку Е.

У файлі test.py реалізована функція, яка виробляє тестування нейронної мережі можна подивитися в додатку И. При запуску скрипта відбувається завантаження вхідних даних - тестових зображень з низькою і високою роздільною здатністю, а також читання бекапу з останнім збереженим станом нейронної мережі. Потім відбувається ініціалізація мережі аналогічно тому, як це реалізовано в файлі train.py.

Після обробки кожної парії тестових зображень відбувається їх збереження в форматі jpg.

Вихідний код даного файлу можна подивитися в додатку Ж.

В обох файлах train.py і test.py використовується допоміжний пакет datetime, за допомогою якого проводиться вимірювання часу, витраченого на навчання або тестування нейронної мережі. У test.py крім цього додатково підключена бібліотека OpenCV для збереження результату роботи нейронної мережі у вигляді jpeg-зображень.

4 ДОСЛІДЖЕННЯ ОТРИМНИХ РЕУЛЬТАТІВ

4.1 Опис експериментів

Навчання нейронних мереж проводилось на комп'ютері з наступною конфігурацією: процесор Intel Core i5-6600 3.30 GHz, відеокарта NVIDIA GeForce GTX 980 6GB, оперативна пам'ять 16 Гб. Обчислення проводилися на відеокарті. Операційна система Windows 10 Home 64-bit.

Для навчання нейронних мереж було використано 5072 кольорових зображень розміру 96 х 96 пікселів. Процес підготовки зображень описаний в попередньому розділі. Кількість зображень, використаних при тестуванні, склала 256. Для навчання та тестування використовувалися різні зображення, тому що це дозволило б у разі виникнення проблеми перенавчання відразу її помітити.

На вхід алгоритмів надходили зображення розміром 24 х 24 пікселя. Ці зображення збільшувалися в 4 рази, тобто до вихідний розмірів 96 х 96 пікселів. Розмір навчального набору (Батч) дорівнював 16. Таким чином, кількість ітерацій, тобто тренувальних наборів, оброблених кожною мережею, склала 317.

Кількість повних проходжень тренувальних наборів для обох нейронних мереж (циклів, «епох», epoch) дорівнювала 200.

Після навчання моделі SRGAN було проведено повторне навчання, але в цей раз після обробки кожного тренувального набору підключався модуль тестування і результат обробки зберігався у вигляді зображення. Це було зроблено для того, щоб мати можливість побачити динаміку поліпшення якості генеруються зображень в міру навчання нейронної мережі.

Приклад такої динаміки представлений на рисунку 4.1. На ньому цифра під зображенням позначає номер циклу навчання.

Рисунок 4.1 – Динаміка навчання SRGAN

Як можна помітити на рисунку 4.1, візуально складно побачити різницю між 100-м і 200-м циклом, однак якщо подивитися на будь-яку з вибраних метрик якості, то на продовженні всього процесу навчання зберігається позитивна динаміка. Показники представлені в таблиці 4.1.

Таблиця 4.1 – Зміна значення якості в процесі навчання

№ цикла	PSNR	SSIM	MSE
1	12,00	0,13	1577,13
10	22,01	0,73	291,47
50	27,01	0,87	125,32
100	27,37	0,89	118,07
200	27,47	0,90	116,02

Динаміка змін показників на прикладі PSNR представлена на рисунку 4.2.

Як видно з рисунка 4.2, швидкість навчання починає значно сповільнюватися вже після 10-го циклу, але тривала динаміка поліпшення показника зберігається аж до 200-го циклу. Після того, як був закінчений процес навчання і тестування нейронний мереж, були згенеровані результати роботи алгоритмів інтерполяції.

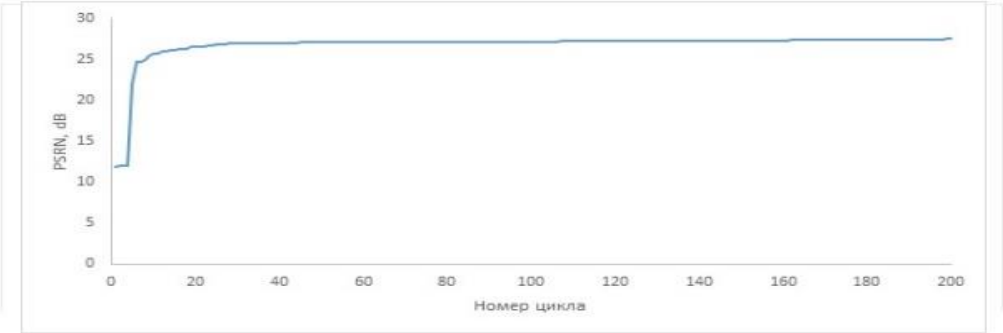


Рисунок 4.2 – Графік зміни показника PSNR в процесі навчання

4.2 Порівняння навчання моделей ШНМ і роботи алгоритмів

Для моделі SRCNN час навчання склав 53 хвилини 26 секунд, тобто середній час на одну повну обробку тестових зображень склав 16 секунд.

У моделі SRGAN навчання зайняло значно більше часу – всього близько 8 години 43 хвилин. Відповідно, середній час на обробку всіх тестових наборів дорівнює 2 хвилинам 37 секундам.

Навантаження на процесор при навчанні було приблизно однакове для обох моделей і становило в середньому 36%. Навантаження на відеокарту в середньому теж було приблизно однакове більшу частину часу роботи мереж і становило 21%.

При проведенні тестування моделей час обробки тестових зображень склав 27 секунд для моделі SRCNN і 33 секунди для SRGAN.

Алгоритм бікубічної інтерполяції і фільтр Ланцоша час на обробку тестових зображень зайняв 0,002005 і 0,561503 секунд відповідно.

Для обох нейронних мереж були використані функції бібліотеки TensorFlow для збереження проміжного або фінального стану. Файл зі збереженим станом нейронної мережі SRCNN становить 441 мегабайт, SRGAN – 490 мегабайт.

4.3 Порівняння результатів роботи алгоритмів і моделей ШНМ за обраних метрик

Були обрані наступні метрики для перевірки: MSE, PSNR і SSIM.

Приклад результату, отриманого в результаті роботи алгоритмів і ШНМ, представлений на рисунку 4.2.

Рисунок 4.2 – Результати роботи алгоритмів

Метрики для цього прикладу наведено в таблиці 4.2.

Таблиця 4.2 – Порівняння результатів на прикладі одного зображення

Метрика	Бікубічна інтерполяція	Фільтр Ланцоша	SRCNN	SRGAN
MSE	774,61	786,71	182,08	115,09
PSNR	19,24	19,17	25,53	27,53
SSIM	0,41	0,40	0,80	0,91

Середні показники результатів обробки всіх зображень представлені в таблиці 4.3.

Таблиця 4.3 – Порівняння середніх показників всіх зображень

Метрика	Бікубічна інтерполяція	Фільтр Ланцоша	SRCNN	SRGAN
MSE	588,61	603,65	200,80	130,25
PSNR	21,26	21,15	26,94	27,63
SSIM	0,57	0,56	0,89	0,92

Як видно з таблиці 4.3, результат роботи моделі SRGAN має найкращі показники. Обидві моделі ШНМ значно перевершують алгоритми інтерполяції, проте між самими моделями відмінність в показниках не така значна.

ВИСНОВКИ

Аналіз сучасних підходів до вирішення завдання поліпшення зображень показав, що незважаючи на факт існування щодо швидких і якісних алгоритмів, постійно з'являються нові методи.

В останні роки все частіше для вирішення цього завдання використовуються штучні нейронні мережі. У більшості досліджень використовуються різні реалізації моделей згорткових нейронних мереж. Однак відносно недавно з'явилися генеративнозмагальні нейронні мережі, які виглядають дуже перспективно.

В рамках даного дослідження була створена програмна реалізація з використанням алгоритмів бікубічної інтерполяції, фільтра Ланцоша, моделей ШНМ SRCNN і SRGAN.

На вибір алгоритмів інтерполяції вплинув той факт, що вони є досить ефективними по співвідношенню часу роботи і якості отриманого результату. Також на вибір вплинуло те, що дані алгоритми вельми широко поширені і використовуються в багатьох додатках.

Вибір моделі SRCNN обгрунтований тим, що дана модель ШНМ є однією з найбільш поширених і перевірених для вирішення завдання поліпшення зображень. Модель SRGAN була обрана через якісні результати роботи. Ця модель є новою і в даний момент не має такої великої кількості програмних реалізацій, як модель SRCNN.

Всі перераховані вище алгоритми і методи було описано.

Програмна реалізація, створена в результаті даного дослідження, дозволяє проводити порівняння не тільки з перерахованими методами, але також дозволяє підключати інші алгоритми і моделі ШНМ.

У цьому дослідженні було описано і створено програмну реалізацію об'єктивних метрик для оцінки результатів роботи алгоритмів збільшення роздільної здатності. Цими метриками є:

- метод середньоквадратичної помилки моделі (Mean Squared Error, MSE);
- пікове відношення сигналу до шуму (Peak Signal-to-Noise Ratio, PSNR);
- індекс структурної подібності (Structural Similarity, SSIM).

При порівнянні результатів за допомогою об'єктивних метрик, можна відзначити, що генеративнозмагальна модель справляється з обробкою зображень ефективніше, ніж згорткова нейронна мережа. Однак для навчання такої моделі потрібно набагато більше часу. Використання навченої моделі генеративнозмагальної мережі для обробки тестових зображень теж займає трохи більше часу, ніж згорткова. Однак результат, який дозволяють отримати обидві ці моделі, значно перевершує алгоритми інтерполяції.

Також варто відзначити такий мінус нейронних мереж, як розмір програмної реалізації. Для обох реалізованих моделей розмір файлу, в якому зберігається стан нейронної мережі, становить понад 400 мегабайт. Для алгоритмів інтерполяції розмір програмної реалізації може становити лише кілька кілобайт.

Повільний час роботи і більший розмір програмної реалізації перешкоджають включенню ШНМ в програмні реалізації для кінцевого користувача. Але результат, який вони дозволяють отримати, перевершує найпоширеніші алгоритми інтерполяції.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. Издание 3-е, исправленное и дополненное. Москва : Техносфера, 2012. 1104 с.

2. Горбачевская Е.Н. Классификация нейронных сетей // Вестник Волжского университета им. В.Н. Татищева. 2012. № 2. С. 128–134.

3. Дорогой Я.Ю. Архитектура обобщенных сверточных нейронных сетей // Вестник Национального техничного университета Украины. 2011. №. 54.

4. Маркелов К.С. Модель повышения информативности цифровых изображений на базе метода суперразрешения. // Инженерный вестник. М. : ФГБОУ ВПО «МГТУ им. Н.Э. Баумана». 2013. № 03. 18 с.

5. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. М. : Телеком, 2013. 384 с.

6. Сазонов В.В. Качественное увеличение деталей изображения // Надежность и качество: тр. Междунар. Симпозиума. 2013. No 1.

7. Суков А.В. Компьютерное зрение. Библиотека OpenCV и ее применение в прикладных задачах робототехники. ГОУВПО «Донецкий национальный технический университет». г. Донецк, 2016.

8. Diederik P., Lei B., Nadav C. A Method for Stochastic Optimization // The Hebrew University of Jerusalem. Advanced Seminar in Deep Learning (#67679). October 18, 2015.

9. Dong C., Loy C.C., Tang X. Accelerating the super-resolution convolutional neural network // European Conference on Computer Vision. Springer. 2016. P. 391–407.

10. Dong C., Loy C.C., He K. Image super-resolution using deep convolutional networks // IEEE transactions on pattern analysis and machine intelligence. 2016. №38(2). P. 295-307.

11. Fadnavis S. Image Interpolation Techniques in Digital Image Processing: An Overview // International Journal of Engineering Research and Applications. 2015. Vol., 4. Issue 10. H.70–73.

12. Goodfellow I., Pouget-Abadie J., Mirza M. Generative adversarial nets // In Advances in Neural Information Processing Systems (NIPS). 2014. P. 2672–2680.

13. Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

14. Kim J., Kwon Lee J., Mu Lee K. Accurate image super-resolution using very deep convolutional networks // In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 1646-1654.

15. LeCun Y., Bottou L., Orr G.B. Efficient BackProp // Neural Networks: Tricks of the Trade. 1998. P. 9-50.

16. Ledig C., Theis L., Huszar F. Photo-Realistic Single Image SuperResolution Using a Generative Adversarial Network // In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

17. Marsland S. Machine Learning: An Algorithmic Perspective. Hoboken : CRC Press, 2011. 406 c.

18. Shi W., Rueckert D., Wang Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network // In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

19. Wang Z., Bovik A.C., Sheikh H.R. Image quality assessment: From error visibility to structural similarity // IEEE Transactions on Image Processing. 2004. vol. 13. no. 4. P. 600—612.

20. Zhou D., Shen X., Dong W. Image zooming using directional cubic convolution interpolation // IET Image Processing. 2012. Vol. 6. No. 6. P. 627-634.

21. Библиотека глубокого обучения Tensorflow. URL : https://habrahabr.ru/company/ods/blog/324898/ (дата звернення 27.11.2019).

22. Generative Adversarial Networks – Hot Topic in Machine Learning. URL : http://www.kdnuggets.com/2017/01/generative-adversarialnetworks-hot-topic-machine-learning.html (дата звернення 23.11.2019).

23. ImageNet – Image database for machine learning. URL : http://www.image-net.org/ (дата звернення 17.11.2019).

24. Implementation of Super Resolution CNN in Tensorflow. URL : https://github.com/tjvandal/srcnn-tensorflow (дата звернення 21.11.2019).

25. NumPy. URL : http://www.numpy.org/ (дата звернення 27.11.2019).

26. OpenCV library. URL : https://opencv.org/ (дата звернення 14.11.2019).

27. Peyman Milanfar.: Enhance! RAISR Sharp Images with Machine Learning. URL : https://research.googleblog.com/2016/11/enhance-raisr-sharp-images-withmachine.html (дата звернення 07.11.2019).

28. Super Resolution using Generative Adversarial Network (SRGAN) URL : https://github.com/tadax/srgan (дата звернення 26.11.2019).

29. The Most Popular Language For Machine Learning and Data Science Is … URL : http://www.kdnuggets.com/2017/01/most-popularlanguage-machine-learning-data-science.html (дата звернення 27.11.2019).