НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«До захисту допущено»
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
з напряму підготовки 6.040303 «Системний аналіз»
на тему: «Згорткові нейронні мережі та їх застосування до розпізнавання дорожніх об'єктів в умовах зашумленості»

Виконав:
студент IV курсу, групи КА-51
Женчук Олексій Валерійович
_____

Керівник:
доцент, к.ф.-м.н. Яковлева А.П. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є. _____

Рецензент:
доцент, к.ф.-м.н. Ільєнко А.Б. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів

без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.040303

«Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ

В.о.завідувача кафедри

_____ О.Л. Тимощук

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Женчуку Олексію Валерійовичу

1. Тема роботи «Згорткові нейронні мережі та їх застосування до розпізнавання дорожніх об’єктів в умовах зашумленості», керівник роботи Яковлева Алла Петрівна, к.ф.-м.н., доцент кафедри ММСА, затверджені наказом по університету від «25»травня 2019 р. №1353с.

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____

_____

4. Зміст роботи _____

_____

_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ _____
(підпис) (ініціали, прізвище)

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

РЕФЕРАТ

Дипломна робота: 101 с., 19 рис., 10 табл., 3 дод., 38 джерел.

ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ЗАДАЧА РОЗПІЗНАВАННЯ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, ЦИФРОВИЙ ШУМ ЗОБРАЖЕННЯ, ЗМЕНШЕННЯ РІВНЯ ШУМУ, РОЗПІЗНАВАННЯ ДОРОЖНІХ ОБ'ЄКТІВ.

Метою даної роботи є дослідження можливості та властивостей застосування згорткових нейронних мереж до задач розпізнавання та класифікації дорожніх об'єктів в умовах наявності цифрового шуму на зображеннях, а також дослідження роботи алгоритмів зменшення рівня шуму в застосуванні до вказаних типів задач.

Актуальність теми: розпізнавання та класифікація дорожніх об'єктів є ключовим для багатьох напрямків, що швидко розвиваються, при цьому рішення на їх основі мають прийматися в реальному часі та виходячи лише з отриманих даних, які можуть мати високий рівень цифрового шуму, що значно ускладнює роботу систем. Тому необхідні моделі, здатні визначати наявність шуму та мати відповідні механізми усунення його впливу.

Об'єктом дослідження є набори цифрових зображень, які містять дорожні об'єкти різних класів і які можуть мати певний рівень цифрового шуму.

Предметом дослідження є моделі згорткових нейронних мереж з оптимізованим використанням параметрів та алгоритми зменшення шуму в зображеннях в їх застосуванні до задачі розпізнавання та класифікації.

Методи дослідження: застосовані моделі згорткових нейронних мереж, методи зменшення рівня шуму в зображеннях, алгоритми навчання нейронних мереж, виконані за допомогою мови програмування Python.

Отримані результати: створено алгоритми розпізнавання та класифікації дорожніх об'єктів з використанням моделей згорткових нейронних мереж та зменшенням рівня цифрового шуму, для чого

реалізовано два варіанти вирішення задачі, для використання в реальному часі та з попередньою обробкою, в ході чого було визначено оптимальний алгоритм для вирішення задачі. За матеріалами бакалаврської роботи готуються до публікації тези доповіді на міжнародній науково-практичній конференції «Інтелектуальні системи та інформаційні технології» (ISIT2019), Одеса, Україна.

ABSTRACT

Thesis: Convolutional neural networks and their application to road objects recognition under noise conditions.

The thesis contains 101 p., 19 fig., 10 tabl., 3 app., 38 sources.

CONVOLUTIONAL NEURAL NETWORK, RECOGNITION PROBLEM, IMAGE CLASSIFICATION, DIGITAL IMAGE NOISE, NOISE LEVEL REDUCTION, ROAD OBJECTS RECOGNITION.

The purpose of this thesis is research of possibility and attributes of applications of convolutional neural networks to the problems of recognition and classification of road objects in conditions with digital noise appearance on images, and also research of efficiency and comparison of noise level reduction algorithms in application to the mentioned problems.

Topicality of the theme: road objects recognition and classification is crucial for many rapidly developing research and application branches, and at the same time the decisions based on the solutions of these problems should be made in real time and only based on the received data, which may have high noise level that makes the systems operations more difficult. Because of this the models that are able to measure the noise level and have noise reduction mechanisms are needed.

The object of research are sets of digital images that contain road objects from different classes and that can have some level of digital noise.

The subject of research are the convolutional neural network models with optimized parameter usage and algorithms of noise level reduction in images in application to the recognition and classification problem.

Methods of research: applied convolutional neural network models, algorithms of noise level reduction in images, neural networks learning algorithms, implemented with the Python programming language.

Received results: the algorithms for recognition and classification of road objects with usage of convolutional neural network models and digital noise level

reduction were created, for what two variants of problem solution were implemented, for real-time application and with data preprocessing, and during that the optimal algorithm for problem solution was determined. On materials of the thesis abstract at the international scientific and practical conference «Intellectual systems and information technologies» (ISIT2019), Odesa, Ukraine, is awaiting publication.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 10
ВСТУП 11
1 ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ ТА ОГЛЯД МЕТОДІВ ЇХ ЗАСТОСУВАННЯ ДО КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ 13
1.1 Вступ 13
1.2 Згорткові нейронні мережі 13
1.3 Операція згортки 17
1.4 Агрегувальні шари 22
1.5 Використання повнозв’язних рівнів 25
1.6 Рівні втрат та вибір функції втрат 26
1.7 Використання активаційних функцій 27
1.8 Висновки до розділу 1 30
2 МОДЕЛІ ЗГЛАДЖУВАННЯ ЗОБРАЖЕНЬ ТА НАЯВНІ НАБОРИ ДАНИХ 32
2.1 Вступ 32
2.2 Цифровий шум в зображеннях 32
2.3 Оцінка рівня шуму 37
2.4 Методи зменшення рівня шуму 39
2.4.1 Методи зменшення рівня шуму: вейвлет-перетворення 41
2.4.2 Методи зменшення рівня шуму: метод повної варіації 44
2.4.3 Методи зменшення рівня шуму: двосторонній фільтр 47
2.5 Постановка задачі класифікації зображень 50
2.6 Набори даних для задачі 51
2.6.1 Набір даних Traffic Signs Dataset 52
2.6.2 Набір даних Graz02 54
2.7 Висновки до розділу 2 55
3 РОЗРОБЛЕНІ МОДЕЛІ ТА РЕЗУЛЬТАТИ ЇХ РОБОТИ 57
3.1 Вступ 57

3.2 Вимоги та інструменти для обробки даних та для задач розпізнавання 57
3.3 Критерії знаходження оптимальної моделі та методу згладжування 59
3.4 Принципи обрання моделей для поставленої задачі 62
3.5 Обрані моделі згорткових нейронних мереж 63
3.5.1 Модель на основі SqueezeNet 63
3.5.2 Модель на основі MobileNet 66
3.6 Принципи застосування методів зменшення рівня шуму до моделей .. 68
3.7 Результати роботи для набору даних Traffic Signs Dataset 70
3.8 Результати роботи для набору даних Graz02 72
3.9 Аналіз результатів для згладжування зображень 73
3.10 Висновки до розділу 3 76
4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ 77
4.1 Вступ 77
4.2 Постановка задачі проектування 77
4.3 Обґрунтування функцій програмного продукту 78
4.4 Обґрунтування системи параметрів ПП 81
4.5 Аналіз експертного оцінювання параметрів 84
4.6 Аналіз рівня якості варіантів реалізації функцій 88
4.7 Економічний аналіз варіантів розробки ПП 89
4.8 Вибір кращого варіанту ПП техніко-економічного рівня 95
4.9 Висновки до розділу 4 96
ВИСНОВКИ 97
ПЕРЕЛІК ПОСИЛАНЬ 99
ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДОПОВІДІ 102
ДОДАТОК Б ПРОГРАМНИЙ ПРОДУКТ НАВЧАННЯ 115
ДОДАТОК В ПРОГРАМНИЙ ПРОДУКТ ШУМ 122

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЗНМ – згорткова нейронна мережа

МЗРШ – методи зменшення рівня шуму

ВПВ – вейвлет-перетворення з використанням методу BayesShrink

ВПV – вейвлет-перетворення з використанням методу VisuShrink

МПВб – метод повної варіації з оптимізацією Бергмана

МПВд – ітеративний метод повної варіації

ДФ – двосторонній фільтр

CNN – Convolutional neural network

ВСТУП

З розповсюдженням технологій прийняття рішень, заснованих на аналізі даних та використанні алгоритмів машинного навчання, велику роль починає відігравати коректне розпізнавання та класифікація об’єктів на зображенні, яке має одночасно забезпечувати необхідну швидкодію для того, щоб результат роботи системи був вчасним та актуальним. Оскільки класичні нейронні мережі погано придатні для роботи з зображеннями через нездатність враховувати просторове та відносне положення об’єктів, ускладненість рівнів, а також через труднощі зі знаходженням набору даних, достатньо великого для того, щоб навчити мережу класичної побудови тощо. Згорткові нейронні мережі значно полегшують роботу з зображеннями з уведенням понять операції згортки та фільтрів, внаслідок чого структура мережі полегшується та залишаються лише важливі зв’язки між нейронами.

Зображення, які створюються за допомогою цифрових камер, можуть страждати від появи на них цифрового шуму через апаратні помилки, різкі несподівані зміни обстановки тощо. Для нейронних мереж поява цифрового шуму може значно знижувати ефективність їх роботи, адже це значно змінює цифрове значення точок на зображенні у порівнянні з точками в їхньому околі, що збиває механізми фільтрів та алгоритми згорткових мереж, знижуючи точність прогнозування. В таких областях застосування, де від правильного визначення класу об’єкта на зображенні залежить прийняте системою рішення, як, наприклад, в безпілотних автомобілях, точність відіграє ключову роль в роботі системи.

Тому в роботі поставлено для виконання наступну задачу:

- провести огляд та аналіз наявних підходів до застосування згорткових нейронних мереж в задачах розпізнавання та класифікації зображень, а також можливі методи та рішення, які для цього використовуються;

- розглянути механізми виникнення цифрового шуму в зображеннях та методи боротьби з ним з метою дослідити вплив алгоритмів зниження шуму на роботу згорткових нейронних мереж при використанні зашумлених зображень;
- визначити моделі та характеристики згорткових нейронних мереж, придатні для застосування до заданого класу задач, та проаналізувати власне роботу згорткових мереж для чистих, зашумлених та відновлених даних та ефективність застосування їх методів в задачах розпізнавання та класифікації дорожніх об’єктів на прикладі вибраних наборів даних;
- на основі отриманих результатів проаналізувати вплив алгоритмів зниження рівня шуму на роботу згорткових нейронних мереж та зробити висновки щодо особливостей та оптимальності їх застосування.

Також в роботі враховується складова ефективної структури згорткових мереж та використання обчислювальних ресурсів з точки зору можливої оптимізації структури мережі для можливості роботи з нею на пристроях зі значно обмеженими ресурсами, де використання надвеликих нейронних мереж неможливе.

РОЗДІЛ 1
ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ ТА ОГЛЯД МЕТОДІВ ЇХ ЗАСТОСУВАННЯ ДО КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1. Вступ

В даному розділі розглядаються основні поняття та методи, пов’язані зі згортковими нейронними мережами та їх застосування до задач класифікації зображень. Описується загальна ідея нейронних мереж, досліджуються причини їх виникнення та поширення в задачах розпізнавання та класифікації зображень, а також відповідні переваги у порівнянні з традиційними нейронними мережами.

Проводиться огляд та опис основних операцій, шарів та методів згорткових нейронних мереж, а також принципи їх побудови та їх відмінності в залежності від поставленої задачі, яку вони вирішують. Розглядаються особливості навчання мереж та пов’язані з цим функції, властивості та параметри. Також описуються гіперпараметри мережі, встановлення яких визначає складність та пристосованість мережі до вирішення тих або інших поставлених задач.

1.2. Згорткові нейронні мережі

Згорткові нейронні мережі (ЗНМ, англ. convolutional neural networks, CNN) – це підвид глибоких нейронних мереж, переважно застосовуваний до розпізнавання об’єктів та класифікації зображень. Згорткові нейронні мережі сприймають та обробляють дані у вигляді тензорів, що дозволяє працювати з даними зображень у природній формі. Кожне вхідне зображення має як параметри ширину та довжину, а також глибину, яка визначається кодуванням зображення. Найбільш поширеним із них є RGB-кодування, в якому колір кожного пікселя на зображенні кодується за допомогою значень

для трьох кольорів – червоного, зеленого та синього. Такі категорії значень, що описують зображення, називаються каналами. Інший вимір тензорів утворюється в ході роботи мережі і містить карти виявлених ознак зображення. Таким чином, згорткові нейронні мережі розглядають кожне зображення як чотиривимірний масив даних. Зазвичай для кожної мережі визначається конкретна необхідна ширина і довжина зображень, хоча також існують мережі, здатні до масштабування.

Для задач розпізнавання в умовах використання великих об’ємів даних необхідно, щоб модель мала високу здатність до навчання та великий відсоток правильних припущень щодо ознак зображення. У порівнянні з традиційними нейронними мережами прямого поширення зі схожою кількістю шарів, згорткові нейронні мережі мають вищу здатність до навчання, оскільки містять набагато менше параметрів та зв’язків. Для традиційних повнозв’язних нейронних мереж для їх правильного навчання в задачах розпізнавання образів необхідна набагато більша кількість даних, оскільки кожне вхідне зображення може мати розмірність у щонайменше кілька сотень тисяч, що вимагатиме відповідну кількість прикладів з різними значеннями для кожного виміру. [1]

Ще однією відмінністю згорткових нейронних мереж є те, що вони зазвичай не вимагають попередньої обробки даних, оскільки використовують дані зображень напряму, що дозволяє спрощувати структуру мережі, зважаючи на упорядкованість початкових даних. Водночас, незважаючи на численні переваги, їх застосування все ще вимагає значної витрати ресурсів, особливо для розв’язання задач із зображеннями з високою роздільною здатністю, тому виникає необхідність розробки згорткових нейронних мереж із високим рівнем оптимізації операцій.

Окрім застосування для задач розпізнавання та задач класифікації зображень, згорткові нейронні мережі завдяки модифікаціям широко застосовуються до задач обробки природної мови (ОПМ, англ. natural

language processing, NLP), зокрема, вони показали високу ефективність в задачах семантичного парсингу, а також моделюванні та класифікації речень. [2]

Структура згорткової нейронної мережі відповідає загальноприйнятій структурі нейронної мережі. Мережа складається з вхідного шару, певної кількості прихованих шарів та вихідного шару. Приховані шари зазвичай складаються зі згорткових шарів, шарів агрегування (субдискретизації), нормалізуючих та повнозв'язних шарів. Ці шари пов'язані між собою шарами з визначеними активаційними функціями. Головним елементом згорткової нейронної мережі є згорткові шари, де до даних з попереднього шару застосовується операція згортки, яка детальніше розглянута в розділі 1.2. [3] Загальна структурна схема побудови згорткової нейронної мережі наведена на рис. 1.1.

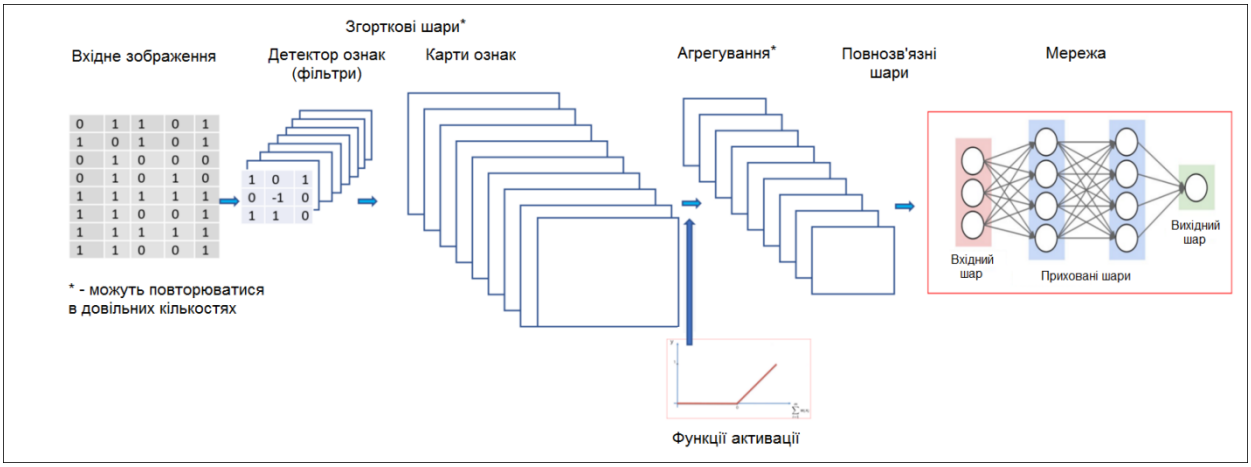


Рисунок 1.1 – Загальна структурна схема побудови згорткової нейронної мережі

Згорткові нейронні мережі здобули широку популярність в задачах розпізнавання зображень завдяки тому, що вони уникають або зменшують головні проблеми, пов'язані з обробкою зображень в традиційних повнозв'язних мережах. Велика кількість зв'язків в таких мережах та велика кількість параметрів у кожному зображенні потребує великої кількості вхідних прикладів, а у випадку їх нестачі швидко виникає перенавчання

мережі, тобто, модель стає занадто складною та навчається на нехарактеристичних другорядних ознаках, знайдених у вхідних даних. Ще більшою проблемою є те, такі моделі не є стійкими до збурень або будь-яких змін у зображеннях, наприклад, до розташування об’єкта в іншій частині зображення або зміні кута погляду на нього. Крім того, такі мережі не здатні до врахування топології вхідних даних: зображення є структурованими даними, де пікселі, що розташовані близько один від одного, мають високу кореляцію, що робить необхідним знаходження локальних ознак та взаємозв’язків для ефективного визначення образів.

В згорткових нейронних мережах ці проблеми вирішуються завдяки використанню кількох ідей, найголовнішою з яких є локальна обробка значень на прихованих рівнях. Кожен з рівнів поділений на частини, кожна з яких сприймає лише дані попереднього рівня, розташовані в певній невеликій області. Такі області називаються локальними рецептивними полями. Використовуючи локальні рецептивні поля, мережа може визначати найпростіші елементи та ознаки зображення, як повороти або грані між ділянками зображення. Іншою ідеєю в основі згорткових нейронних мереж є спільні ваги, використання яких зменшує чутливість мережі до змін положення об’єктів на зображенні, незначних поворотів та його спотворення: так, детектор певної простої ознаки зображення, що використовується на певній його ділянці, може визначати аналогічні ознаки і на іншій його ділянці. Завдяки цьому детектори для локальних рецептивних полів на різних ділянках будуть мати однакові ваги, і, як наслідок, в різних частинах зображення для схожих ознак виконуються однакові перетворення.

Крім того, важливим елементом є операція субдискретизації, яка дозволяє зменшувати загальну розмірність даних, при цьому зменшуючи їх чутливість до значних збурень. Зазвичай субдисретизація в архітектурі мережі чергується із операціями згортки, таким чином поступово зменшуючи розмірність даних та збільшуючи кількість карт ознак.

1.3. Операція згортки

Операції згортки виконуються у згорткових шарах нейронної мережі та є найголовнішим їх елементом. Власне згортка є математичною операцією, яка за допомогою дії ядра згортки (детектор ознак, англ. kernel або feature detector) k на вхідне зображення у вигляді тензора x отримує як результат карту ознак m (англ. feature map). Кожний детектор ознак складається з деякої визначеної кількості фільтрів (англ. filters), кожний з яких в свою чергу є тензором параметрів з розмірністю, яка як правило дозволяє покривати лише невелику частину зображення в ширину та висоту, але при цьому обов'язково має відповідати розмірності в глибину. Такі фільтри і є параметричною частиною нейронної мережі, до якої застосовується навчання. Окрім виявлення ознак та переходів, фільтри також можуть використовуватися як інструменти збільшення різкості чи розмивання зображення.

Крім того, власне кількість фільтрів також може бути одним із гіперпараметрів нейронної мережі, оскільки збільшення кількості фільтрів веде до збільшення кількості потенційно виявлених ознак та закономірностей, але при цьому занадто велика їх кількість може призвести до повторень та знаходження ознак, які не є характеристичними, тобто до ускладнення та потенційного перенавчання моделі. Таким чином, кількість фільтрів відповідає кількості карт ознак, яка визначає глибину вихідних даних, а, отже, і глибину вхідних для наступного рівня мережі, що відповідно визначає кількість необхідних параметрів і на наступному кроці.

В ході навчання фільтри набувають можливості визначати ознаки зображення, які й групуються в карти ознак. Кожен фільтр відповідає за створення однієї карти ознак. Карта ознак формується завдяки тому, що фільтр переміщується зображенням та визначає ознаки для сусідніх

рецептивних полів. Крок переміщення s (англ. stride) може відрізнятися і є одним з параметрів моделі. Таким чином, хоча абсолютне положення кожної ознаки на зображення на карті і не зберігається, але це не має значення, оскільки відносне положення ознак одна відносно одної залишається сталим, і, таким чином, властивості зображення зберігаються.

В загальному вигляді операція може бути записана математично так:

$$m(t) = (x * k)(t) = \sum_{a=-\infty}^{a=+\infty} x(a)k(a+t),$$

Значення кожного детектора ознак за межами визначеної області його дії вважаються нульовими, тому нескінченну суму можна замінити на скінченну зі значеннями в заданій області. [4] Використовуючи властивість комутативності для операції математичної згортки, отримуємо спрощену функцію, придатну для застосування в нейронних мережах та створення детекторів для згорткового шару, та яка споріднена з взаємокореляційною функцією для значень зображення:

$$M(i,j) = (K * X)(i,j) = \sum_{m}\sum_{n} K(m,n)X(i-m, j-n),$$

де $M(i,j)$ – елемент карти ознак з координатами i та j,

X – вхідне зображення,

K – детектор ознак,

m, n – розмірності детектора ознак.

Приклад фільтру для операції згортки для двовимірного випадку наведений на рис. 1.2.

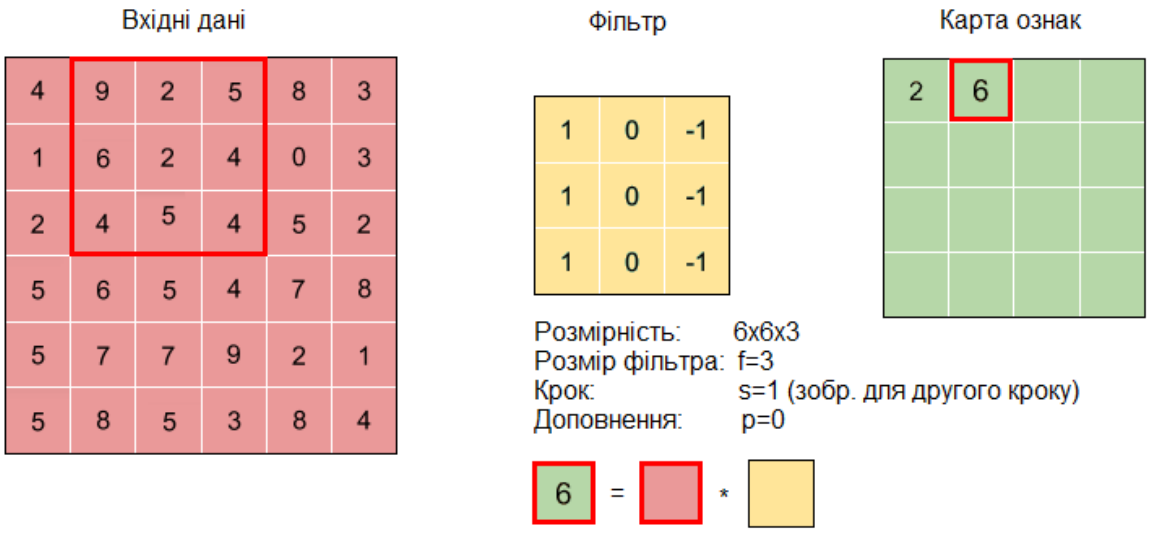


Рисунок 1.2 – Приклад одновимірного фільтру для операції згортки при розмірності зображення 6x6 з трьома каналами, розміру фільтру 3, розміром кроку 1 та відсутності доповнення нулями

Таким чином, кожний елемент карти ознак можна інтерпретувати як результат перетворення за допомогою скалярного добутку значень з деякої області (рецептивного поля нейрона), який пов’язаний з сусідніми значеннями з тієї ж карти.

Ознаки, які здатні виявляти фільтри, відрізняються в залежності від рівня мережі, на якому вони розташовані. Для перших рівнів це можуть бути прості переходи кольору або зміни освітленості, на глибших рівнях фільтри можуть виявляти складні кути, закруглення або образи. Це можливо завдяки використанню на наступних рівнях результатів згорток на попередніх та збільшенні кількості точок початкового зображення, які впливають на кожне значення карти ознак на більш глибоких рівнях. Ця ж властивість збільшує вплив певної ознаки на результат при її ближчому розташуванні до центра зображення, оскільки значення, що належать до такої ознаки, будуть впливати на більшу кількість значень окремо взятої карти ознак, ніж ознака біля краю зображення за рахунок охоплення більшою кількістю рецептивних полів фільтрів.

Крок обходу зображення (англ. stride) s є одним із гіперпараметрів мережі, тобто параметром, що має бути визначений ще процесу навчання, на етапі конструювання моделі. Збільшення кроку веде до зменшення розмірності вихідних даних (а, отже, до зменшення складності моделі) та зменшення накладання рецептивних полів для різних значень.

Доповнення нулями (англ. padding) p є ще одним гіперпараметром і може використовуватися в тих випадках, коли небажано зменшувати розмірність карт ознак у порівнянні з вхідними даними, а також загалом її контролю. Воно полягає в розширенні вхідного зображення визначеною кількістю нульових значень з кожного краю. Застосування доповнення нулями дозволяє також збільшувати вплив значень зображення, які розташовуються з його краю, оскільки без його використання крайні значення використовуються в картах ознак лише один раз, тобто, втрачається інформація з країв зображення. Результат застосування доповнення нулями на 1 шар для прикладу з рисунка 1.2 наведено на рис. 1.3.

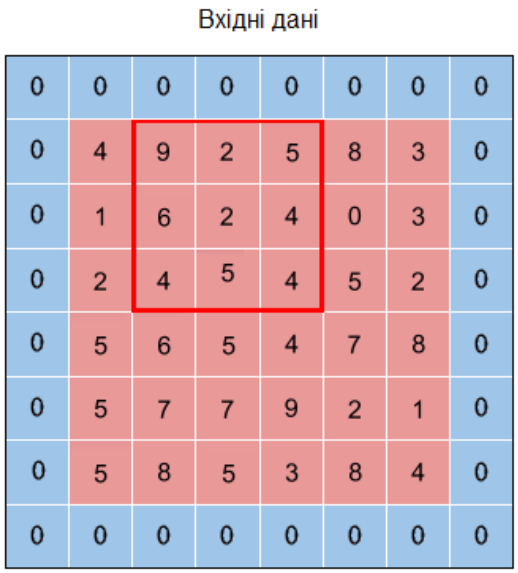


Рисунок 1.3 – Приклад застосування доповнення нулями на 1 шар

Розмірність фільтрів визначається з розрахунком, що з її збільшенням фільтри знаходитимуть більші ознаки, які зазвичай зустрічаються рідше, але

можуть бути вагомішими [5]. В сучасних дослідженнях існує тенденція до її зменшення, оскільки це дозволяє відповідно зменшити використання ресурсів, так як кількість параметрів у фільтрах зі збільшенням розмірності зростає квадратично. Для збереження розмірів рецептивних полів у таких випадках використовується вертикальне послідовне розташування згорток. Це також дозволяє покращити роботу мережі, оскільки таким способом збільшується її глибина, замість ширини для згорток великої розмірності. Інколи використовуються навіть згортки з розмірністю 1 з метою зменшення глибини даних перед виконанням операцій з великими витратами ресурсів.

Розмірність карт ознак залежить від вказаних гіперпараметрів та може бути обчислена наступним чином:

$$d_m = \frac{d_x - f + 2 * p}{s + 1},$$

де d_m – розмірність карт ознак (ширина або висота),

d_x – відповідна розмірність вхідних даних,

f – розмірність фільтрів,

p – доповнення нулями,

s – розмір кроку обходу зображення.

Також можливе паралельне використання фільтрів з різною розмірністю на одному рівні. Такий вид згорток називається груповими згортками (англ. grouped convolution) та вимагає групування вихідних даних для кожної розмірності окремо від інших груп, що вимагатиме також застосування різних згорток до цих груп на наступних рівнях. Це збільшує ефективність мережі, оскільки кожна група згорток виконує функцію окремого згорткового шару, та широко використовується в сучасних архітектурах.

Крім того, інколи застосовується операція розширення (англ. dilation), яка полягає в застосуванні елементів фільтру не до сусідніх елементів зображення, а до елементів з визначеним кроком. Це дозволяє значно зменшити розмірність даних, якщо вони містять багато надлишкової інформації, а також узагальнити ознаки. [5]

Після застосування фільтрів та виконання необхідних операцій серед описаних вище, до отриманих даних виконується додавання зсуву b, який також є одним із параметрів мережі. Всі виконані перетворення є лінійними, тому для створення нелінійних змін у даних застосовується функція активації, властивості якої детальніше розглянуті в розділі 1.5. Загальний процес перетворень у згортковому шарі може бути описаний рівнянням:

$$y = f(w * x + b),$$

де y – вихідний результат обчислень на згортковому шарі,

w – детектор ознак,

x – вихідні дані,

b – зсув.

1.4. Агрегувальні шари

Агрегувальні шари або шари субдискретизації (англ. pooling layers) є одним з основних структурних елементів згорткових нейронних мереж, як і згорткові шари. Такі шари можуть бути як глобальними, так і локальними, тобто розповсюджуватись тільки на окремі групи вхідних даних, які будуть рецептивними полями для нейронів поточного шару. Головною задачею агрегувальних шарів є зменшення розмірності даних з одночасним збереженням найважливіших характеристик шляхом формування залежності між кількома елементами (нейронами) з попереднього шару з єдиним

елементом даного шару. Тому при побудові мережі вони зазвичай використовуються з певною періодичністю між згортковими шарами. [6] Слід зазначити, що агрегувальні шари зберігають глибину вхідних даних, при цьому значно запобігаючи перенавчанню.

Агрегувальні шари бувають двох підтипів: усереднювальні (англ. average) та максимізаційні (англ. maximal). Інколи використовуються також мінімізаційні шари (англ. minimal) або агрегувальні шари за L_2-нормою. Усереднювальні шари надають кожному елементу середнє значення серед тих, що належать до відповідної йому групи нейронів з попереднього шару, тоді як максимізаційні надають максимальне значення. Формально операції для обох типів агрегування записуються наступним чином:

$$p(i,j) = \frac{\sum_{m,n} x(i-m, j-n)}{m * n},$$

для усередненого агрегування,

$$p(i,j) = \max_{i,j}\big(x(i-m, j-n)\big),$$

для максимизаційного агрегування,

де $p(i,j)$ – значення елементу поточного рівня з координатами i та j,

x – вхідні дані з попередніх рівнів,

m, n – розмірність рецептивного поля.

Важливою функцією агрегувальних шарів є збільшення стійкості мережі до збурень в даних – наприклад, до зміни кута погляду на об’єкт або зміни його положення. [7] Завдяки вибору значення з множини за певним визначеним критерієм (усереднювальним або максимізаційним), зміна вхідних даних на незначні значення мало впливає на результати роботи агрегувального рівня. Таким чином, схожі ознаки, наприклад, переходи, кути

або заокруглення будуть мати схожі вихідні значення, незважаючи на деякі зміни в положенні або освітленні. Краще в такій задачі себе виявляє максимізаційне агрегування.

Агрегувальні шари мають декілька однакових гіперпараметрів з операцією згортки: так, мають бути визначені крок обходу зображення, який часто обирається так, щоб накладання уникалося; доповнення нулями, а також розмірність рецептивного поля, хоча в переважній більшості мереж використовується вікно розмірністю 2x2. [8] Агрегувальні шари здатні значно зменшувати розмірності вхідних даних та вплив окремих елементів з невеликими значеннями на результат, тож використання такого малого рецептивного поля виправдане. Існують також архітектури мереж, які не використовують агрегувальні шари взагалі, замінюючи їх згортковими шарами зі збільшенням кроку. [6]

Розмірність результуючих даних для агрегувальних шарів може бути вирахувана з використанням значень гіперпараметрів аналогічно до обчислення розмірності карт ознак для згорткових рівнів.

Інколи замість операції агрегування використовується операція субдискретизації (англ. subsampling), [7] яка полягає в знаходженні суми елементів для рецептивного поля кожного з нейронів, після чого ця сума множиться на параметр з наступним додаванням параметра зсуву та застосуванням нелінійної функції активації. Формалізовано таку операцію можна записати так:

$$p(i,j) = f\left(w * \sum_{m,n} x(i-m, j-n) + b\right),$$

де $p(i,j)$ – значення елементу поточного рівня з координатами i та j,

x – вхідні дані з попередніх рівнів,

m, n – розмірність рецептивного поля.

1.5. Використання повнозв’язних рівнів

Повнозв’язними рівнями (англ. fully-conneccted layers) в згорткових нейронних мережах називають такі рівні, де всі нейрони з наступного шару поєднані зв’язками з нейронами попереднього шару, як і у більшості шарів у звичайних нейронних мережах. Використання таких рівнів на початкових та прихованих рівнях мережі невиправдане, адже воно ускладнює модель і навіть може ігнорувати знайдені раніше за допомогою згорток та агрегації ознаки та закономірності. Проте, повнозв’язні рівні зазвичай використовуються на передостанньому кроці роботи мережі для підготовки знаходження результатів на виході мережі.

Для роботи повнозв’язного рівня елементи матриць та шарів з попереднього кроку випрямляються в послідовність значень. Такі шари, як і згорткові, виконують обчислення скалярного добутку даних та параметрів з додаванням зсуву. Тому загальна формула повнозв’язного рівня схожа на формулу згорткового:

$$y = f(w^T * x + b).$$

З відмінністю, що w позначає не детектор ознак, а масив параметрів рівня.

На повнозв’язних рівнях застосовують функцію активації, результат якої або буде використовуватися на наступному рівні, або слугувати для обчислення результатів мережі на виході. Тому типи активаційних функцій, які використовуються на цьому рівні, відрізняються від тих, що використовуються на згорткових та агрегаційних, та бувають різними в залежності від положення рівня та загальної мети мережі. Так, одними з

найпопулярніших функцій активації для формування виходів є Softmax функція та функція сігмоїди.

Повнозв’язні рівні можуть розглядатися як згорткові рівні з розмірністю детектора ознак 1x1, де реалізовані всі зв’язки. [9]

1.6. Рівні втрат та вибір функції втрат

Рівень втрат (англ. loss layer) є зазвичай останнім рівнем мережі та відповідає за оновлення параметрів при її тренуванні. На цьому рівні за допомогою попередньо визначеної функції втрат (англ. loss function) вираховується різниця між справжніми значеннями тренувальних даних (справжніми даними) та результатами роботи мережі (прогнозом), після чого при наступному етапі тренування мережі ці дані використовуються для застосування алгоритмів тренування.

Вибір та задання функції втрат є одним з найважливіших елементів побудови нейронної мережі, адже від цього значною мірою залежить як швидкість та якість тренування мережі, так і здатність мережі оцінювати точність отримуваних результатів. Значення функції втрат є критерієм оцінки ефективності роботи мережі, адже за її допомогою всі переваги, недоліки та помилки мережі мають бути зведені до одного значення, яке має характеризувати мережу таким чином, щоб результати роботи для різних варіантів моделі можливо було впорядкувати за точністю. Крім того, функція втрат має враховувати особливості конкретної задачі, для якої використовується мережа.

Двома головними типами функцій втрат для нейронних мереж є функції перехресної ентропії (англ. cross-entropy functions) та функції середньоквадратичної похибки (англ. mean-squared error functions). [10] Проте, для згорткових мереж в задачах класифікації для зіставлення результатів роботи мережі з одним з визначених класів використовуються

імовірності того, що вхідні дані належать кожному з визначених класів. В такій задачі знаходження параметрів відбувається за принципом максимальної правдоподібності, оскільки мережа за допомогою наближення до правильних параметрів намагається наблизити розподіл результатів до розподілу справжніх даних. Таким чином, шуканим критерієм помилки роботи мережі буде різниця між імовірностями належності вхідних даних до всіх класів, отриманих мережею та заданих у вибірці. Тому, для задач класифікації зі згортковими мережами використовуються переважно тільки функції перехресної ентропії. [11] Більше того, функція середньоквадратичної похибки для задачі класифікації може бути підвидом функції перехресної ентропії, якщо дані розподілені за нормальним законом.

Функція перехресної ентропії, яка використовується як функція втрат для задачі класифікації, має назву функції логарифмічних втрат (англ. log-loss function) та застосовується паралельно з використанням функцій активації Softmax (для багатокласової класифікації) та сігмоїди (для бінарної класифікації) на останньому рівні мережі для обчислення прогнозів. [9] Вона може бути знайдена за формулою:

$$L(p, y) = -\sum_{n} y_n \ln p_n \, , n \in [1, N],$$

де p – отримані за допомогою мережі ймовірності належності даних до вибраних класів,

y – справжні значення для вхідних даних,

N – кількість класів.

1.7. Використання активаційних функцій

Функції активації або активаційні функції (англ. activation functions) використовуються як нелінійний фактор після згорткових, агрегаційних, а інколи й інших шарів згорткових нейронних мереж, формуючи їх остаточні вихідні дані. Вибір та використання функцій активації має великий вплив на роботу мережі, оскільки без їх застосування мережа не мала б змоги навчатися розпізнавати нелінійні закономірності та розподіли. Крім того, якщо рівні мережі не мають функції активації між собою, буде утворюватися лінійна залежність між початковим та кінцевим рівнями такого проміжку, що фактично робить значення кількості рівнів між ними невпливовим, роблячи ці рівні еквівалентними одному. Також за допомогою правильного вибору активаційної функції можна контролювати, наскільки великий вплив на результати буде мати елемент (нейрон) в залежності від його значення, навіть до заборони впливу деяких із них.

Зазвичай функції активації зменшують значення даних або мають меншу за діапазон вхідних даних, часто обмежену область значень, оскільки в інакшому випадку є ризик нескінченного зростання параметрів. Крім того, вони мають бути диференційовними для того, щоб уможливлювати алгоритм зворотнього поширення помилки, та, відповідно, градієнтні методи навчання мережі. Вибір функції активації значною мірою впливає на те, яким чином необхідно задавати початкові значення для мережі перед початком її навчання: так, якщо функція активації не наближує тотожне перетворення поблизу нульової точки, то при заданні випадкових невеликих значень мережа може гірше навчатися.

Випрямлена лінійна одиниця (англ. rectified linear unit, скорочення ReLU) є функцією активації, яка все частіше застосовується в сучасних нейронних мережах. Вибір саме цієї функції зумовлений тим, що вона оброблює дані та навчає мережу в кілька разів швидше за інші функції активації, при цьому також створюючи нелінійність. Це, окрім пришвидшення навчання мережі, ще знижує можливість її перенавчання,

оскільки мережа навчається більш складним зв’язкам за однаковий термін, у порівнянні з іншими функціями активації, що в свою чергу дозволяє навчати складніші моделі на тих самих даних. [13] Формула знаходження значення ReLU:

$$y = f(x) = \max(0, x),$$

де y – елемент поточного рівня,

x – елемент з вхідних даних.

Як слідує з формули функції, вона усуває негативні значення з мережі, надаючи їм нульового значення, натомість виконуючи тотожне перетворення для інших значень. Швидкість роботи цієї функції з апаратної точки зору покращується ще й тим, що вона є звичайним порівнянням значення з нулем. Похідна ReLU дорівнює 0 для від’ємних значень і 1 для всіх інших, тому робота алгоритму зворотнього поширення помилки також спрощується.

Тим не менше, оскільки ReLU не наближує тотожного перетворення поблизу нульової точки, вона вимагає наявності хоча б одного додатного початкового значення для ефективного навчання. З іншого боку, активація не всіх початкових значень позитивно впливає на роботу мережі через ефект розрідження. Впровадження ReLU було пов’язано зі знайденням аналогії з її роботою в обробці візуальної інформації корою головного мозку. [9]

Ще двома функціями, які традиційно використовуються для функцій активації в нейронних мережах, є гіперболічний тангенс ($f(x) = \tanh(x)$) та функція сігмоїди, яка крім того застосовується як функція активації на останньому рівні мереж в задачах бінарної класифікації, оскільки її область значень [0;1] та це дозволяє апроксимувати імовірності належності вхідних даних до двох класів:

$$y = f(x) = \frac{1}{1 + e^{-x}}.$$

В задачах багатокласової класифікації на останньому рівні мережі зазвичай використовується функція Softmax завдяки тому, що вона враховує всі значення з попереднього рівня мережі та може обрахувати відповідну імовірність для кожного класу. Саме ця функція застосовується для формування результатів в цій роботі:

$$y_i = f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \forall i, j \in (1, N),$$

де y_i – елемент поточного шару з індексом i,

x_i – елемент попереднього шару з індексом i,

N – кількість елементів на попередньому шарі.

Фактично, функція Softmax нормалізує екпоненційні значення даних з попереднього рівня, тим самим переводячи вхідний масив довільних значень в область [0;1], що й використовується для обчислення імовірностей. Функція може бути модифікована шляхом заміни бази степені з e на інше число, тим самим збільшуючи вагу великих значень на попередньому рівні (при збільшенні бази), або навпаки зрівнюючи вплив різних значень (при зменшенні бази). Крім того, функція стійка до збільшення всіх показників з попереднього рівня на стале значення, збільшуючи стійкість результуючих імовірностей належності до класів.

1.8. Висновки до розділу 1

В даному розділі було розглянуто згорткові нейронні мережі, один з підвидів нейронних мереж, який показав себе ефективним в задачах

класифікації та розпізнавання зображень, але також може використовуватися і для інших цілей. Основними їх елементами є згорткові рівні з операцією згортки, які розпізнають ознаки на зображення, агрегувальні рівні з операціями, які узагальнюють зображення та забезпечують покращену стійкість моделі, повнозв’язні рівні для обробки результатів, отриманих на згорткових та агрегувальних, рівні втрат, що забезпечують оновлення параметрів моделі при навчанні та покращення її роботи, та функції активації, які забезпечують нелінійність системи та роблять можливими багатошарову побудову мереж, а також розпізнавання складних характеристик.

Згорткові нейронні мережі навчаються завдяки градієнтним алгоритмам, що змінюють параметри моделей в напрямку зменшення помилки. Для задач класифікації, які розглядаються в цій роботі, результатом роботи згорткової нейронної мережі є імовірності належності вхідного зображення до кожного з вибраних класів.

РОЗДІЛ 2
МОДЕЛІ ЗГЛАДЖУВАННЯ ЗОБРАЖЕНЬ ТА НАЯВНІ НАБОРИ ДАНИХ

2.1. Вступ

В цьому розділі розглядається цифровий шум зображення та методи зменшення зашумленості (згладжування, англ. denoising) зображень, описуються особливості їх застосування для задачі класифікації, постановка якої також робиться у цьому розділі.

Також описуються обрані набори даних з категоріями дорожніх об’єктів, які будуть використовуватися для застосування на них відповідних моделей в задачах класифікації. Розглядаються методи роботи з цими даними для визначення та контролю рівня цифрового шуму.

2.2. Цифровий шум в зображеннях

Шум є випадковими коливаннями показників елементів зображення, наприклад, значень кольорових каналів чи яскравості, які не належать до реального зображення та є його спотвореннями. Шум у зображеннях може виникати як при їх обробці через технічні причини, так і через дію зовнішніх факторів. Найчастішими умовами, які спричиняють появу шуму, є процес зйомки, а саме фіксації та збереження зображення у камері, а також його передача каналами зв’язку. З розповсюдженням цифрових технологій шум, що виникає завдяки їх використанню, отримав назву цифрового.

Цифровий шум буває двох типів: просторово залежний та просторово незалежний. Рівень та значення просторово залежного шуму залежать від координат точок на зображенні, і він зазвичай виникає при власне процесі зйомці зображення. Прикладом просторово залежного шуму є періодичний шум, який виникає через перешкоди електромеханічного характеру або

некоректні зміни експозиції та створює на зображенні періодичні структури з більш світлих, темних або просто спотворених точок. Просторово незалежний шум зустрічається частіше, не має кореляції зі значеннями пікселів та має випадковий характер. Два типи шуму мають значні відмінності в частоті випадків, причинах та умовах виникнення, а також способах знешумлення та властивостях, тому вони як правило розглядаються окремо та мають дуже різні способи їх зниження [14]. Тому в цій роботі розглядається саме просторово незалежний шум як такий, що зустрічається найчастіше в звичайній фотографії і має природні причини виникнення.

Найчастішим типом цифрового просторово незалежного шуму є гаусівський шум - такий шум, функція щільності імовірності якого відповідає нормальному (гаусівському) розподілу:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

де $x \in (-\infty; +\infty)$ – рівень шуму,

μ – середнє значення,

σ – середнє квадратичне відхилення.

Гаусівський шум зустрічається в незначній кількості в будь-якому сигналі та широко використовується для моделювання та наближення інших типів шуму. Його причини можуть бути дуже різними та виникати через дію багатьох факторів одночасно. Так, сенсори, що сприймають зображення, завжди мають певний рівень шуму через те, що рівень освітленості для них визначається кількістю фотонів, які потрапили на сенсор, що сама по собі є випадковим числом.

Такий тип шуму, виникнення якого зумовлюється дискретною природою світла, називається дробовим шумом. Дробовий шум розподілений за законом Пуассона, проте через велику кількість окремих подій

(потрапляння фотонів на сенсор) він майже завжди апроксимується гаусівським розподілом, окрім значень з надзвичайно низькою інтенсивністю. В практичному застосуванні типи шуму, розподілені за законом Пуассона, завжди апроксимуються моделлю гаусівського шуму через велику кількість подій у них та відповідній збіжності до гаусівського розподілу. [15] [16] Власна температура сенсорів, а також електронні системи, що передають сигнал у камері, також вносять власний рівень шуму, який прямо моделюється гаусівським розподілом. Рівень шуму може значно підвищуватися при низькому освітленні.

Всі ці види шуму або моделюються як гаусівський, або наближуються ним та моделюються як гаусівський на практиці. [17] Єдиним виключенням із цього правила є мікрофотографія, де через низьку кількість подій потрапляння фотонів на сенсори шум часто не піддається апроксимуванню, особливо на темних ділянках, хоча все рівно алгоритми зменшення шуму, зроблені з припущенням щодо його гаусівського розподілу, будуть зменшувати його рівень, лише не так ефективно. [18] Крім того, моделювання рівня шуму гаусівським розподілом можливе тільки наближено, оскільки для шуму неможливі такі значення, які б виводили значення пікселів за рамки встановленого інтервалу значень кольорів, тобто умова $x \in (-\infty; +\infty)$ на практиці не є здійсненною, хоча абсолютна більшість значень шуму все рівно буде відрізнятися від середнього значення не більше ніж на значення 3σ.

Окрім гаусівського, іншими типами просторово незалежного шуму є рівномірний шум (шум квантування), що виникає як помилка переводу аналогового вхідного сигналу в цифровий; шум «сіль та перець» (англ. salt and pepper noise), що є імпульсним типом шуму та виникає при конвертуванні та передачі зображення і представляється у вигляді окремих непов’язаних світлих та темних точок; шум Релея; експоненційний та гамма-шум. Проте, ці види шуму або зустрічаються в конкретних спеціальних

областях зйомки зображень (шуми гамма, експоненційний та Релея зустрічаються в лазерних, діапазонних або рентгенівських зображеннях), або надзвичайно мало зустрічаються та впливають на дані (рівномірний шум), або представляють окремий випадок неадитивного шуму та все одно непогано апроксимуються гаусівським розподілом (шум «сіль та перець»). [14] [19] Тому для моделювання шуму та для більшості алгоритмів знешумлення на зображеннях використовується саме гаусівський шум через його властивість наближення шумів з іншими розподілами.

Загалом механізм виникнення та зниження рівня шуму можна формалізовано проілюструвати наступною схемою (рисунок 2.1). Вхідне зображення $I_c(x, y)$ без цифрового шуму піддається дії компоненти шуму $n(x, y)$, внаслідок чого утворюється зашумлене зображення $I_n(x, y)$. Така модель називається адитивною та Для операції зниження рівня шуму відомими вхідними даними є $I_n(x, y)$, а також деяка інформація про шум $n(x, y)$, наприклад, його розподіл або визначений рівень. Вихідними даними для операції є знешумлене зображення $\overline{I_c}(x, y)$, яке буде наближенням до початкового $I_c(x, y)$. Тим не менше, слід зазначити, що оцінити точність наближення, тобто різницю між зображеннями, без наявності початкового в реальних умовах неможливо, тому в якості критерія точності результатів зниження рівня шуму використовуються остаточні результати роботи моделей, розглянутих в розділі 3 роботи, тим більше, що точність наближення може й не характеризувати придатність алгоритму знешумлення до роботи з моделлю розпізнавання. Покращити наближення зазвичай можна шляхом збільшення інформації про шум $n(x, y)$, а також за допомогою вибору правильного методу зниження рівня шуму.

Рисунок 2.1 – Схема виникнення та зниження рівня цифрового шуму на зображенні

Таким чином, найпопулярнішою моделлю представлення шуму є адитивна модель, яка, зокрема, використовується для гаусівського шуму:

$$I_n(x,y) = I_c(x,y) + n(x,y)$$

де $I_n(x,y)$ – зашумлене зображення,

$I_c(x,y)$ – початкове зображення,

$n(x,y)$ – компонент шуму.

Інколи в цій моделі враховується також зниження якості зображення в сенсі розмиття, тобто вплив на зображення функції розмиття $h(x,y)$, яка спотворює зображення. Це рівняння тоді виглядає так:

$$I_n(x,y) = h(x,y) * I_c(x,y) + n(x,y)$$

де $I_n(x,y)$ – зашумлене зображення,

$h(x,y)$ – функція розмиття,

* - математична операція згортки,

$I_c(x,y)$ – початкове зображення,

$n(x,y)$ – компонент шуму.

Моделі зниження рівня шуму працюють тільки над очищенням рівня власне шуму, при цьому зазвичай не впливаючи на початкове розмиття та не

покращуючи його, тож поняття розмиття стосується власне початкового зображення. Крім того, згорткові нейронні мережі, які будуть застосовуватися для розпізнавання розглядуваних зображень, містять власні механізми боротьби з розмиттям зображень, як, наприклад, агрегувальні шари мережі. Тому в цьому випадку можливе розмиття не враховується як характеристика якості початкового зображення, яка до того ж може бути дещо усунена за допомогою моделей мереж.

Слід зазначити, що для задач розпізнавання та класифікації зображень необхідні великі набори даних з сотнями прикладів для кожного класу. Сучасні дослідники, за відсутності відповідних наборів даних з такою великою кількістю зображень одного класу, які б утворилися шляхом природніх помилок та утворення шуму, використовують генерацію гаусівського шуму для того, щоб створити відповідні набори. Відомі набори даних для зображень з шумом, присутнім на них від початку, зазвичай дуже малі та містять різноманітні зображення з багатьох сфер життя. [20] Тому навіть для розробки алгоритмів шуму використовуються зашумлені зображення, отримані додаванням гаусівського шуму на початкові зображення. [19] [21]

2.3. Оцінка рівня шуму

Оцінка рівня шуму є необхідною для правильної побудови та роботи алгоритмів зменшення рівня шуму. Його вимірювання в загальному випадку є складною задачею, оскільки власне вміст чистого зображення може бути надзвичайно різним і містити в собі неочікувані піки значень, переходи між яскравими та темними ділянками, нестандартні природні особливості зображуваних об’єктів тощо. Ця особливість так само впливає і на дію методів зниження рівня шуму, розглянутих в наступному розділі. В абсолютній більшості реальних випадків, чисте (реальне) зображення

невідоме, тому оцінка рівня шуму, як і його зменшення, має робитися, виходячи з припущень щодо характеру шуму. Описана апроксимація шуму для зображень як гаусівського значно допомагає в цьому.

Оцінити рівень для шуму, який наближено моделюється гаусівським розподілом, можна за допомогою оцінки середнього квадратичного відхилення для зображення, оскільки розподіл шуму повністю задається цією характеристикою та багато алгоритмів зниження шуму використовують саме її як основний параметр для застосування відповідних методів.

Середнє квадратичне відхилення можна вирахувати за методом, запропонованим Д. Доного та І. Джонстоном у 1992 році [22]. За цим методом, зображення у формі значень точок для всіх каналів кольорів піддається дискретному вейвлет-перетворенню (перетворенню за допомогою застосування вейвлета – коливальної функції, яка прямує до нуля в обох напрямках, але при цьому має відхилення коливального характеру поблизу нульової точки), після чого знаходиться середнє абсолютне відхилення для деталізуючих коефіцієнтів (коефіцієнтів при компоненті вейвлет-функції, англ. detail coefficients) отриманого перетворення. Приклад застосування вейвлет-перетворення до зашумленого зображення наведено на рис. 2.2. Середнє абсолютне відхилення пов’язане з шуканим середнім квадратичним відхиленням пропорційним зв’язком з коефіцієнтом, що для гаусівського розподілу є константою та дорівнює зворотному значенню до значення оберненої функції розподілу в точці 0,75. Виходячи з цього рівняння і проводиться обрахунок середньоквадратичного відхилення. Дискретне вейвлет-перетворення було застосоване у вигляді багатовимірного перетворення за допомогою вейвлетів Добеші з двома зникаючими моментами.

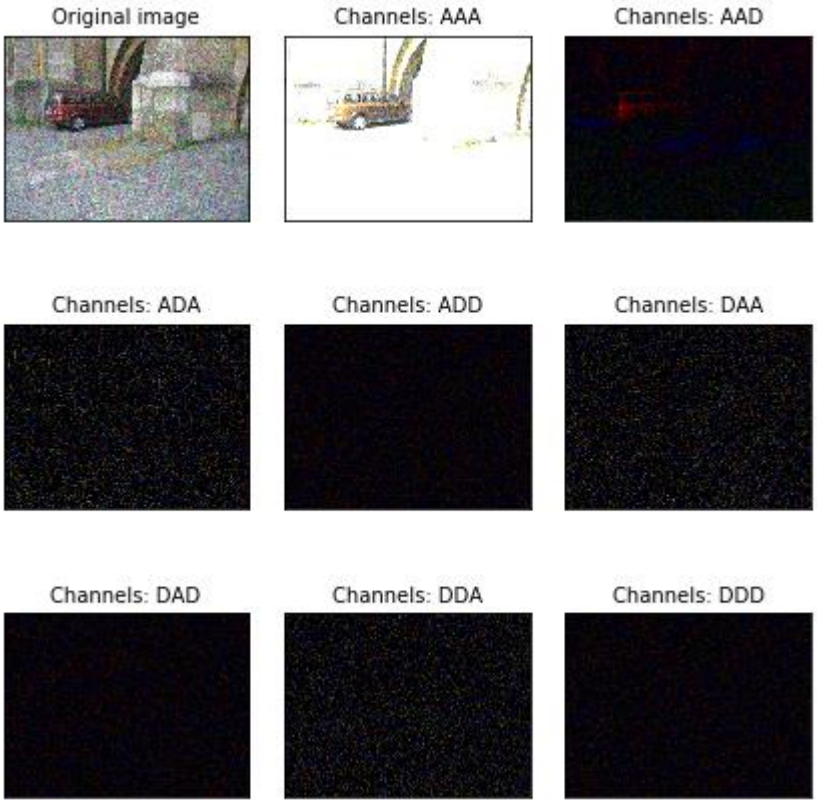


Рисунок 2.2 – Приклад застосування вейвлет-перетворення до зашумленого зображення, результат для деталізуючих коефіцієнтів на усьому зображенні, за допомогою якого знаходиться рівень шуму, у правому нижньому куті

2.4. Методи зменшення рівня шуму

Зменшення рівня шуму (знешумлення, згладжування, англ. denoising) має на меті відновити чисте зображення у випадку, якщо на ньому присутні зашумлені компоненти. Зменшення рівня шуму зазвичай засновується на відомостях або припущеннях щодо того, який саме тип спотворення подіяв на зображення та намагається наблизити реалізацію зворотного процесу, оскільки володіючи інформацією про характеристики спотворень, є можливим оцінити початкове зображення, як це показано у формулі в попередньому розділі. Ефективне зменшення рівня шуму може скласти значний вплив на результати роботи алгоритмів розпізнавання, оскільки буде

забезпечено коректний розподіл та природнє розміщення вхідних даних, що полегшить їх використання.

Процес зменшення рівня шуму ще називається процесом згладжування, оскільки часто його застосування зводиться до наближення екстремальних значень точок з великою різницею з оточенням до менших наближених значень в різних формах та різними способами. [23] Тому найпростіші, часто лінійні, моделі зниження рівня шуму полягали в заміні значень точок на середні значення точок навколо, проте такі методи також значною мірою згладжують ознаки та переходи зображення. Такі побічні ефекти негативно впливають і відновлене на зображення як таке, але для застосування технологій розпізнавання та згорткових нейронних мереж переходи та грані на зображеннях мають ключове значення. Тому як ціль знаходження методу зниження шуму будемо вважати такі методи, які б зберігали (а в найкращому випадку ще й відновлювали) більшість характеристик зображення.

Зазвичай, такі методи є нелінійними та враховують структурні особливості зображення, при цьому зберігаючи такі особливості, як переходи та кути недоторканими. [19] Серед них можна виділити методи, засновані на диференціальних рівняннях, як, наприклад метод повної варіації. Ще одним з методів, який здобув визнання після поширення згаданих у попередньому розділі вейвлет-перетворень, є зниження рівня шуму за допомогою його застосування та використання порогових значень. Метод на основі двостороннього фільтра знайшов значне застосування, оскільки попри порівняну простоту, він використовує не тільки відстань між точками для їх усереднення, а ще й різницю між їх характеристиками, що і дозволяє мати таку властивість.

2.4.1. Методи зменшення рівня шуму: вейвлет-перетворення

Завдяки структурі вейвлет-перетворення, за його допомогою можна оцінити рівень шуму на кожному шарі зображення, що робить можливим відповідне налаштування алгоритму зменшення рівня шуму таким чином, що його дія була направлена на прибирання тільки шуму, без значних втрат до інформаційної складової зображення, і, що найважливіше для цієї роботи, без значного впливу на ознаки чи переходи в ньому. Метод, як і всі описувані тут, розрахований на роботу з найпоширенішим видом шуму, адитивним шумом, який і розглядається як модель шуму в зображенні.

Суть методів зменшення рівня шуму з використанням вейвлет-перетворення полягає в тому, що застосовується алгоритм знаходження порогових значень вейвлет-коефіцієнтів, які відповідають за частину сигналу, а в даному випадку зображення, без шуму. Якщо як базовий вейвлет обрати вейвлет із властивістю ортогональності, то при перетворенні вхідний білий шум буде відображатися також у білий шум. Таким чином, вейвлет-перетворення від шумного зображення може вважатися зашумленим варіантом чистого вейвлет-перетворення. [24]

У вейвлет-перетворенні для чистого зображення без шуму більшість деталізуючих коефіцієнтів будуть близькими до нуля та не нестимуть інформації про сигнал. Але у випадку зображення з присутнім на шумом всі із коефіцієнтів матимуть шум з однаковим середньоквадратичним відхиленням, як це показано у [22]. Отже, для отримання чистого сигналу необхідно виділити такі значення, які б перевищували рівень шуму, а значення з відповідно низьким рівнем будуть вважатися чистим шумом. Для цього і вводиться поняття порогових значень. Порогові значення зазвичай застосовуються лише до деталізуючих коефіцієнтів перетворення, оскільки вони відповідають умовно високочастотним компонентам сигналу, тоді як апроксимуючі відповідають переважно умовно низькочастотним і містять

більшість інформації про сигнал (зображення). Очищення зображення від шуму відбувається шляхом надання нульового значення всім коефіцієнтам, значення яких нижче за порогове.

Визначення порогового значення λ є найважливішим етапом налаштування роботи цього методу. Загалом є два підходи до зниження шуму за цим алгоритмом: м'який та важкий. За м'яким алгоритмом, значення коефіцієнтів після прибирання шуму стискаються до нуля:

$$\eta_S(\omega, \lambda) = \mathrm{sgn}(\omega)\,(|\omega| - \lambda),$$

де $\eta_S(\omega, \lambda)$ – коефіцієнт після застосування порогових значень,

ω – відповідний початковий коефіцієнт,

λ – порогове значення.

Тоді як для важкого алгоритму тільки значення коефіцієнтів, менших за порогове, прирівнюються до нуля:

$$\eta_H(\omega, \lambda) = \omega I_{|\omega| \geq \lambda},$$

де $I_{|\omega| \geq \lambda}$ – індикатор того, що попереднє значення за модулем більше за порогове.

Вибір власне порогового значення також відбувається за алгоритмом, яких було в різний час запропоновано декілька, але з них було обрано для застосування та порівняння алгоритми BayesShrink та VisuShrink. В обох із цих алгоритмів застосовується оцінене значення середньоквадратичного відхилення шуму, отримане за методом, описаним у попередньому пункті.

За методом VisuShrink, порогове значення визначається як $\sigma\sqrt{2\ln n}$, де σ – визначене середньоквадратичне відхилення шуму, n – розмірність вхідних даних. Цей метод своєю перевагою має те, що він уникає утворення

невеликих знижень значень точок з відновленими значеннями у порівнянні з точками навколо. [22]

Метод BayesShrink використовує баєсів ризик з того припущення, що середньоквадратичне відхилення власне вейвлет-коефіцієнтів, які наближаються нормальним розподілом, наближається ним. [25] Знаходження порогового значення відбувається за наступною формулою:

$$\lambda = \frac{\sigma^2}{\sqrt{\max(\frac{\sum_{i,j=1}^{n} Y_{i,j}^2}{n^2} - \sigma^2, 0)}},$$

де λ – порогове значення,

σ – визначене середньоквадратичне відхилення шуму,

n – розмірність вхідних даних,

Y – вхідні дані.

Вираз $\frac{\sum_{i,j=1}^{n} Y_{i,j}^2}{n^2}$ представляє середньоквадратичне відхилення для даних зашумленого зображення.

Після завершення очистки зображення від шуму за допомогою порогового значення відбувається зворотне вейвлет-перетворення, в результаті чого отримується зображення зі зниженим рівнем шуму.

Приклад зменшення рівня шуму за допомогою вейвлет-перетворення та використання методів BayesShrink та VisuShrink наведено на рис. 2.3.

Рисунок 2.3 – Приклад зменшення рівня шуму з застосуванням вейвлет-перетворення з алгоритмами вибору порогового значення BayesShrink та VisuShrink для двох зображень

2.4.2. Методи зменшення рівня шуму: метод повної варіації

Метод зменшення рівня шуму, заснований на мінімізації повної варіації зображення, можна розділити на два алгоритми: метод повної варіації, заснований на розділеній оптимізації Брегмана та ітеративний метод повної варіації для багатовимірних зображень. Обидва алгоритми реалізують принцип, на якому заснований метод повної варіації: елементи зображення з шумом мають великі значення варіації, а, отже, її зменшення призведе до наближення зображення до його чистої форми. При цьому метод не розмиває чи змінює границі та форми зображення.

Розділена оптимізація Брегмана була запропонована у 1991 році як метод оптимізації для L1-регуляризованих просторів, а саме вирішення задач випуклої оптимізації для функцій, визначених в гільбертовому просторі з використанням відстані Брегмана, але застосування в зниженні рівня шуму знайшла лише у 2012 році як метод при знаходженні повної варіації. [26]

Механізм знешумлення зображення за допомогою методу повної варіації з’явився раніше, і його головною ідеєю було вирішення нескінченновимірної задачі оптимізації для знаходження відновленого зображення, наближаючи відновлене зображення до початкового.

В цьому методі використовується не власне алгоритм Брегмана, а розділений алгоритм Брегмана, який є його варіантом, пристосованим до вирішення недиференційовних випуклих задач оптимізації, та розроблявся також з урахуванням можливого застосування до задач зменшення рівня шуму в зображеннях. [27] Алгоритм є ітеративним, і вирішується за оптимізаційним методом чергування напрямків зі зміною значення показників u (відновленого зображення), для знаходження необхідного наближення реального зображення, та d (наближення градієнта u), для збереження точності наближення градієнта, з вирішенням власне оптимізаційного рівняння методом Гаусса-Зейделя. Кінцевою метою алгоритму є знаходження наближення найменшого можливого значення повної варіації u як суми абсолютних значень градієнта для всіх елементів зображення. Ця задача розділяється як оптимізація суми значень наближень градієнта d та квадратичної різниці між реальним зображенням та u, та переводиться в форму, необхідну для застосування алгоритму Брегмана з введенням додаткового параметра b, що відповідає за оновлення наближення d. Досягнення результату перевіряється оцінкою квадрата різниці між значеннями u на поточній та попередній ітерації. Загальна формула мінімізації для методу може бути записана таким чином:

$$\underset{d}{\operatorname{argmin}} \sum_{i,j} \left|d_{i,j}\right| + \frac{\lambda}{2} \sum_{i,j} \left(f_{i,j} - u_{i,j}\right)^2 + \frac{\gamma}{2} \sum_{i,j} \left|d_{i,j} - \nabla u_{i,j} - b_{i,j}\right|^2,$$

де $d_{i,j}$ – наближення градієнта u,

f – зашумлене зображення,

u – відновлюване зображення,

λ – параметр методу,

γ – штрафний параметр, стале значення, задається на початку алгоритму,

b – додатковий параметр для збереження наближення d до ∇u на кожному кроці алгоритму.

Другий алгоритм для зменшення рівня шуму за методом повної варіації був запропонований А. Шамболь у 2004 році. За цим методом, відбувається пошук відновленого зображення за формулою:

$$\|u - f\|^2 = N\sigma^2,$$

де u – відновлюване зображення,

f – зашумлене зображення,

N – кількість точок на зображенні,

σ – оцінка середньоквадратичного відхилення для зображення.

Пошук та оцінка відбувається наступним чином: спершу припускається, що значення $\sqrt{N}\sigma$ знаходиться на інтервалі $\left(0; \left\|f - \bar{f}\right\|\right)$, де $\bar{f}$ – середнє значення точок зашумленого зображення. Після цього відбувається пошук такого значення параметра λ, для якого $\|\pi_{\lambda K}(f)\| = \sqrt{N}\sigma$, тобто задача зводиться до пошуку проекції зашумленого зображення на стиснену в λ разів множину K, яка складається з таких елементів, всі числа яких за модулем менші за 1. Ця проекція шукається ітеративно після задання початкового значення λ_0 шляхом його оновлення за відповідністю до вказаного вище рівняння:

$$\lambda_{n+1} = \frac{\sqrt{N}\sigma}{\left\|\pi_{\lambda_n K}(f)\right\|}\lambda_n,$$

де λ_i – значення коефіцієнта λ на i-тому кроці алгоритму.

Таким чином знаходиться оцінка відновленого зображення завдяки прирівнюванню різниці між зображеннями до знайденої за способом вище проекції. Збіжність цього методу до відновленого зображення доведена в [28]. Цей метод використовує менше обчислень, ніж перший варіант, та внаслідок цього є швидшим за нього, що є перевагою для його використання при роботі з зображеннями в режимі реального часу.

Приклад зменшення рівня шуму методом повної варіації за алгоритмами оптимізації Брегмана та ітеративного на основі двох зображень наведено на рис. 2.4.

Рисунок 2.4 – Приклад зменшення рівня шуму за методом повної варіації з використанням алгоритму Брегмана та ітеративного алгоритму для двох зображень

2.4.3. Методи зменшення рівня шуму: двосторонній фільтр

Метод зменшення рівня шуму за допомогою двостороннього фільтра має як переваги відносну простоту обчислень, збереження всіх ознак зображення та використання для роботи просторове розташування та схожість за значеннями для різних точок. Двосторонній фільтр використовує

нелінійну комбінацію близьких як за положенням, так і за характеристиками значень до кожної точки для відновлення зображення, що й надало йому таку назву, а також працює одночасно з усіма каналами зображення для контролю правильності їх обробки в кожній точці. [29] Саме врахування схожості між різними значеннями дозволяє методу зберігати всі характерні ознаки. Технологія двостороннього фільтра завдяки вказаним перевагам знайшла значне застосування на практиці, наприклад, часто саме вона використовується для обробки зображення в різних комерційних додатках та програмах.

Для знаходження вагів для схожості значень точок зображення зазвичай використовуються гаусівські функції, хоча теоретично може бути обраний будь-який спосіб їх знаходження, виходячи з інформації про зображення та необхідний тип метрики. [30] Таким чином, знаходження відновленого зображення за білатеральним фільтром може бути записане так:

$$I_{new}(p) = \frac{1}{W_p} \sum_{q \in S} I(q) c_{\sigma_s}(\|p - q\|) s_{\sigma_r}(\|I(p) - I(q)\|),$$

де I_{new} – відновлене зображення,

I – початкове зашумлене зображення,

p – поточна точка на зображенні, що розглядається, центр фільтра,

q – інша точка на зображенні, яка розташовується у межах вікна фільтра S,

s_{σ_r} – функція, що визначає подібність за характеристиками для точок p та q,

c_{σ_s} – функція, що визначає геометричну подібність точок p та q,

W_p – нормуючий параметр, що визначається так:

$$W_p = \sum_{q \in S} c_{\sigma_s}(\|p - q\|) s_{\sigma_r}(\|I(p) - I(q)\|)$$

Завдяки правильному вибору функцій для визначення подібності між точками можливо осягнути того, щоб в областях з переходами значень (грані, кути тощо) значення функції для точок з подібного боку грані були близькими до одиниці, а для точок з протилежного нульовими. Для гаусівського шуму ці функції будуть мати такий вигляд:

$$f_{\sigma_i}(p, q) = e^{-\frac{1}{2}\left(\frac{\|d(p) - d(q)\|}{\sigma_i}\right)^2}$$

де σ_i – параметри відповідно геометричного та фотометричного поширення для різних функцій,

$d(p)$ – координати точки для геометричної подібності та характеристики точки для фотометричної подібності.

Роль параметрів σ_i полягає у визначенні області S, де діє операція фільтрування. Крім того, параметр σ_s рекомендується прив’язувати до визначеного рівня шуму. [30] Таким чином, ваги при значеннях точок зменшуються, якщо вони розташовуються далеко від тієї, яка розглядається, або їх значення значно відрізняється. Слід зазначити, що білатеральний фільтр має багато спільних рис з описаною в розділі 1 операцією згортки, яка використовується в згорткових нейронних мережах.

Приклад зменшення рівня шуму з використанням двостороннього фільтра наведено на рис. 3.5.

Рисунок 2.5 – Приклад зменшення рівня шуму за методом використання двостороннього фільтра для двох зображень

2.5. Постановка задачі класифікації зображень

Задача класифікації зображень полягає в тому, щоб за вхідним зображенням A визначити, чи належить об’єкт на цьому зображенні до деякого класу c_j з визначеної множини класів $C = \{c_i,\ i \in (1, \dots n)\}$, де n – кількість класів. Часто ця задача зводиться до того, що визначаються імовірності $P = \{p_i, i \in (1, \dots n), \sum_{i=1}^{n} p_i = 1, p_i > 0\}$ належності зображення A до кожного класу $c_j \in C$, після чого рішення про те, чи належить зображення до певного класу приймається на основі визначеного критерію обрання класу.

Таким критерієм може критерій Топ-1 (англ. Top-1), в якому обирається клас з найбільшою отриманою імовірністю, або, для задач класифікації з великою кількість класів, критерій Топ-5 (англ. Top-5), де як результат роботи вважаються 5 класів з найбільшими імовірностями. За іншим критерієм як результат обирається множина класів, імовірність належності зображення до яких більша за деяке задане значення p_{fix}.

Існують задачі контрольованої класифікації (англ. supervised classification) та неконтрольованої, або спонтанної класифікації (англ. unsupervised classification). При контрольованій класифікації на заданій вибірці попередньо визначаються області, які належать до необхідних класів, або, у випадку задачі класифікації зображень, визначається належність до них елементів (зображень) вибірки. Для такої задачі характеристики класів відомі ще до навчання мережі. Водночас для задачі неконтрольованої класифікації множина класів та характеристики вибірки невідомі, і необхідне визначення ознак елементів з їх подальшою кластеризацією та знаходження результуючих класів шляхом визначення кластерів об’єктів.

В даній роботі, виходячи зі специфіки предметної області – дорожніх об’єктів, де наперед відомі класи типи об’єктів, що зустрічаються, а також з прийнятої структури наборів даних для задач класифікації об’єктів, розглядається задача контрольованої класифікації, тому у наборах даних, що будуть розглядатися в наступних розділах, визначена належність кожного елементу до класу з заданої множини.

2.6. Набори даних для задачі

Для реалізації роботи описаних у попередніх розділах методів зниження рівня шуму та побудови моделей згорткових нейронних мереж для задачі розпізнавання дорожніх об’єктів, а також для їх застосування на реальних даних, було обрано два набори даних, які представляють зображення з різних категорій дорожніх об’єктів, поєднаних у класи. Кожен з наборів представляє собою збірку розміром у кілька тисяч зображень, які були зроблені на дорогах у відповідних країнах світу.

2.6.1. Набір даних Traffic Signs Dataset

Набір даних Traffic Signs Dataset було презентовано у 2011 дослідниками з Лінчепізького університету. [31] Набір даних було створено шляхом встановлення на автомобіль цифрової камери та запису даних на 350 кілометрах автодоріг Швеції, як в населеній зоні, так і поза нею, та на автодорогах різного значення. Через такий спосіб збору інформації розподіл даних за категоріями знаків має наближати справжній розподіл знаків, які зустрічаються на шляху автомобіля під час руху. Камера була направлена в напрямку руху зі зміщенням в праву сторону, задля кращого охоплення знаків з правої частини дороги. Зображення для знаків робилися з різної відстані як кадри з відеозапису, зробленого камерою.

Загалом, набір даних має близько 20000 зображень, проте з них лише 20% були пристосовані саме для задачі контрольованої класифікації, тоді як для решти зображень не були присвоєні відповідні дані щодо їх класів. Записи для кожного зображення з пристосованих даних містять інформацію про їх клас, ступінь видимості знаку (розмитий чи чітко зображений), та те, чи знак належить до поточної дороги, чи до її відгалуження. В контексті задачі класифікації мають значення тільки записи про клас зображення.

Набір має 7 класів найпопулярніших дорожніх знаків: «Пішохідний перехід», «Об’їзд перешкоди з правого боку», «Зупинку заборонено», «Обмеження максимальної швидкості 50 км/год», «Обмеження максимальної швидкості 70 км/год», «Головна дорога» та «Дати дорогу».

Загалом, отриманий набір даних для задачі класифікації містить 2193 зображень. Кількість зображень, які належать до кожного з представлених класів, наведено в таблиці 2.1.

Таблиця 2.1 – класи зображень для набору даних Traffic Signs Dataset та кількість прикладів для кожного із них

Клас дорожнього знаку на зображенні	Кількість прикладів
«Пішохідний перехід»	118
«Об’їзд перешкоди з правого боку»	383
«Зупинку заборонено»	130
«Обмеження максимальної швидкості 50 км/год»	139
«Обмеження максимальної швидкості 70 км/год»	268
«Головна дорога»	917
«Дати дорогу»	238

Приклади зображень, які входять до кожного з класів набору даних, наведено на рис. 2.6.

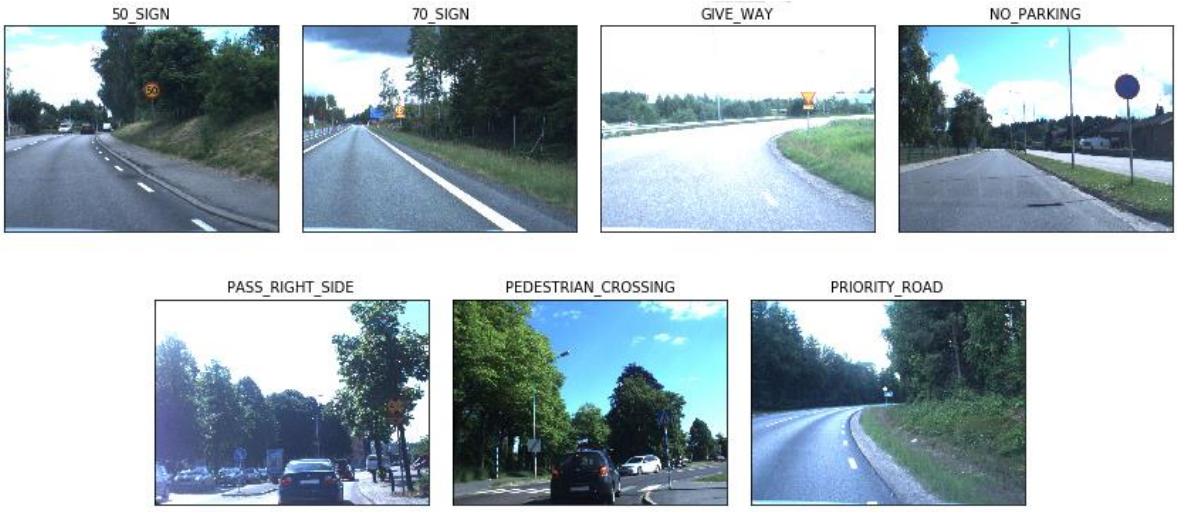


Рис. 2.6 – приклади зображень, які входять до кожного із класів набору даних Traffic Signs Dataset

2.6.2. Набір даних Graz02

Набір даних під назвою Graz02 (або GRAZ-02) було створено дослідниками з Технічного університету Граца у 2004 році саме як набір даних для задачі розпізнавання та класифікації об’єктів. [32] Початковою метою його створення був збір зображень зі складними реальними характеристиками, де об’єкти класів з’являються на зображеннях в ситуаціях з великою кількістю інших об’єктів задля уникнення ситуації, коли об’єкт з’являється у схожих положеннях, наприклад, при зйомці на автостраді.

Всього набір складається з чотирьох категорій основних учасників дорожнього руху: автомобілів, велосипедів, пішоходів та четвертого класу, який представляє всі інші об’єкти.

Особливостями набору даних є велика відмінність зображень для кожного представленого класу, тобто, наприклад, велика кількість різних положень та моделей автомобілів на зображеннях, а також різноманітність у фонових деталях зображення, тобто з великою кількістю ситуацій, в яких зображені відповідні елементи класів, а також різними положеннями цих об’єктів на зображенні. При цьому, в наборі даних для усіх класів зустрічаються схожі ситуації задля урівноваження їх можливого впливу на результати.

Загалом набір містить 1476 зображень, належність яких до різних класів показана в таблиці 2.2.

Таблиця 2.2 – класи зображень для набору даних Graz02 та кількість прикладів для кожного із них

Клас об’єкта на зображенні	Кількість прикладів
Велосипед	365
Автомобіль	420
Пішохід	311
Клас для сторонніх об’єктів	380

(зображення, що не містять ні велосипедів, ні автомобілів, ні пішоходів)	

Приклади зображень, які входять до кожного з класів, наведено на рис. 3.7.

Рисунок 2.7 – Приклади зображень, які входять до кожного із класів набору даних Graz02: велосипеди, автомобілі, пішоходи та сторонні об’єкти

2.7. Висновки до розділу 2

В даному розділі було спершу розглянуте поняття цифрового шуму, визначені основні причини та ситуації його виникнення, розглянуто типи цифрового шуму та зв’язки між ними, а також механізми його моделювання та наближення різних видів шуму моделями, зокрема, показано наближення гаусівським шумом інших для шуму в зображеннях. Визначено алгоритм оцінки рівня шуму за допомогою вейвлет-перетворення, яке згодом використовується в методах зменшення рівня шуму.

Було визначено три методи зменшення шуму, які будуть використовуватися для зменшення рівня шуму, при цьому у двох з них описано по два варіанти методів. При виборі методів було обґрунтовано необхідність підбору методу таким чином, щоб він забезпечував необхідний рівень знешумлення, при цьому не впливаючи на основні ознаки та характеристики зображення, важливі для роботи згорткових нейронних мереж та результатів зниження рівня шуму загалом.

Також була зроблена постановка задачі класифікації, в рамках якої будуть застосовуватися описані методи, та представлені два набори даних з різними категоріями реальних дорожніх об’єктів, в тому числі дорожніх знаків та учасників дорожнього руху, за допомогою яких буде реалізовуватися робота описаних методів та алгоритмів на реальних даних з різними ситуаціями.

РОЗДІЛ 3
РОЗРОБЛЕНІ МОДЕЛІ ТА РЕЗУЛЬТАТИ ЇХ РОБОТИ

3.1 Вступ

В даному розділі розглядаються та описуються моделі згорткових нейронних мереж, розроблені для задачі розпізнавання дорожніх об’єктів в умовах зашумленості для двох наборів даних, описаних у розділі 2. Описуються інструменти та продукти, необхідні для роботи з моделями, шумом на зображеннях та обробки наборів даних.

Визначаються критерії обрання оптимальної моделі та оптимального методу знешумлення зображень. Описується архітектура моделей, а також результати їх роботи для різних варіантів даних та різних методів усунення шуму. Отримані результати аналізуються та порівнюються між собою, на можна зробити висновки щодо оптимальних моделей для цих задач.

3.2. Вимоги та інструменти для обробки даних та для задач розпізнавання

Розробка архітектур моделей, їх навчання, робота з наборами даних, шумом в зображеннях, використання алгоритмів знешумлення та знаходження результатів роботи моделей здійснювалося в таких умовах та за допомогою:

- мови програмування Python покоління Python 3, версія мови 3.6;
- середовище Jupyter Notebook з подальшим запуском моделей, моделюванням роботи програм та обробкою результатів в системі Google Colaboratory – хмарній системі, створеній для роботи з дослідженнями в областях глибокого та машинного навчання з можливістю використання віддаленого сервера з графічним процесором, яка дозволяє запускати файли для Jupyter Notebook

віддалено та фактично є інтерпретатором з запуском коду на віддаленій машині;

- навчання моделей та робота програми в системі Google Colaboratory задля ефективної обробки даних здійснювалася на віддаленому сервері, наданому системою в режимі онлайн. Доступні віддалені сервери обладнані графічним процесором з лінійки NVIDIA Tesla T4 з об’ємом пам’яті 14,8 Гб;
- доступ до системи Google Colaboratory здійснювався за допомогою веб-браузера Google Chrome версії 71;
- створення моделей було проведено в операційній системі Linux Ubuntu 18.04 LTS, з процесором Intel Core i3 2,13GHz та графічним процесором Nvidia GeForce 512 Мб.

Тим не менше, використання Google Colaboratory для відтворення роботи програмного продукту не є обов’язковим, оскільки головною метою її використання в роботі було пришвидшення обчислень та більш ефективне навчання моделі, зважаючи на великий обсяг пам’яті та обчислювальний ресурс, необхідний для обробки та навчання моделей класифікації зображень. Проте, при використанні системи Google Colaboratory наведені в наступному абзаці технічні та інструментальні вимоги непотрібні, єдиною вимогою для відтворення моделювання в такому випадку буде наявність актуальної версії веб-браузера.

Таким чином, технічні та інструментальні вимоги для відтворення результатів моделювання та роботи з шумом на комп’ютері:

- процесор Intel Pentium або Intel Core, бажано з частотою більше 2.0GHz;
- мова програмування Python 3.6 або вище;
- середовище Jupyter Notebook для запуску програми;
- операційна система Ubuntu, Windows або MacOS, але для коректної роботи серед версій Windows бажано використовувати Windows 10;

- вільний простір більше за 6 Гб на диску задля розміщення наборів даних, програмного файлу та збереження результатів роботи моделей;
- бібліотеки для мови програмування Python, вказані нижче.

Для створення програмного продукту були використані наступні бібліотеки мови Python:

- os: робота з файлами, папками та операційною системою, створення файлів;
- tensorflow: побудова моделей нейронних мереж та робота з алгоритмами машинного навчання, контроль інформації щодо використовуваного процесора, підтримка бібліотеки keras;
- keras: побудова моделей згорткових нейронних мереж, їх навчання та знаходження результатів їх роботи;
- numpy: ефективна робота з даними у вигляді масивів, генерація випадкових даних;
- matplotlib: побудова графіків та візуалізація зображень;
- skimage: ефективна робота зі звичайними зображеннями та зашумленими зображеннями;
- pywt: робота з вейвлет-перетвореннями та їх коефіцієнтами.

3.3. Критерії знаходження оптимальної моделі та методу згладжування

Серед запропонованих архітектур моделей та методів згладжування зображень за задачею необхідно знайти ті, які б з наборами даних найкраще підходили до її вирішення. Як було вказано в розділі 1.6, критерієм оцінки ефективності роботи мережі є значення функції втрат після кожної ітерації навчання моделі. Також, значення функції втрат на різних етапах навчання моделі демонструє його ефективність та показує відсутність або присутність перенавчання. Проте, функція втрат оцінює ефективність роботи в процесі навчання та вказує на величину розбіжності між отриманими та справжніми

даними, але не показує відносну кількість даних, для яких мережа робить правильні висновки щодо класу, що дуже важливо для оцінки правильності вирішення задачі класифікації. Для цього використовується поняття точності (англ. accuracy).

Точність є метрикою оцінки результатів задачі класифікації та позначає частину даних, щодо яких модель зробила правильні висновки. [33] Точність набуває значень з проміжку [0;1] та визначається за відношенням:

$$a = \frac{\sum_i c_i}{\sum_i n_i}, i \in [1, N],$$

де a – точність,

n_i – загальна кількість зроблених припущень для i-го класу,

c_i – кількість зроблених правильних припущень для i-го класу,

N – кількість визначених класів.

Слід зазначити, що краще (нижче) значення функції втрат не гарантує краще (вище) значення точності. [34] Модель може давати давати значення, дуже близькі до дійсних (до 0 або 1 для задачі класифікації) для певних значень, і неточні для інших, тим самим створюючи мале значення функції впливу, але цих дуже близьких значень буде мала кількість, тож модель буде правильно визначати лише невелику кількість даних, тож точність буде низькою. З іншого боку, модель може не бути дуже впевненою в отриманих значеннях, тобто розбіжність між дійсними та отриманими даними буде досить великою, проте ці значення будуть достатніми, щоб відношення до класів для більшості даних визначалася правильно. Тому в якості міри оптимальності роботи моделі використовується саме точність.

Для ефективного навчання мережі та перевірки правильності її роботи вибірка наявних даних d розбивається на дві підвибірки: тренувальну вибірку d_{train} та валідаційну вибірку d_{val}. Таке розбиття необхідне, адже при навчанні

моделі лише за даними з однієї вибірки вона може почати перенавчатися або почати виявляти ознаки, які дозволяють покращувати роботу моделі лише для даних з тестової вибірки, а не для такого класу даних загалом. Тому якщо перевірка роботи алгоритму та оцінка його ефективності робиться не на тій же вибірці, що й навчання, це дозволяє отримати її більш незалежну оцінку та перевірити роботу моделі для класу даних загалом. [35] Точність визначення даних на валідаційній вибірці називається валідаційною точністю (англ. validation accuracy).

Інколи, при відсутності довіри до результатів роботи на валідаційній вибірці, робиться виділення ще однієї вибірки d_{test}, яку називають тестовою та перевірка точності моделі на якій проводиться вже після остаточного завершення її навчання та приймається як міра її ефективності. Таке розбиття з іншого боку зменшує кількість даних для тренувальної та валідаційної вибірок, тож в цій роботі для набору даних Graz02 використовується лише розбиття на дві вибірки, зважаючи на те, що кількість даних в наборах даних є не надто великою з точки зору глибокого навчання, і навчання мережі та перевірка точності її роботи стає дуже складною при таких малих об’ємах даних. Таким чином, навчання та визначення ефективності роботи моделі проводиться за допомогою наступних кроків:

1. Навчання моделі на тренувальній вибірці d_{train};
2. Перевірка точності моделі на валідаційній вибірці d_{val};
3. Зміна гіперпараметрів моделі, виходячи з валідаційної точності;
4. Повторення кроків 1-3 до досягнення бажаного результату в кращому випадку, або припинення покращення моделі при застуванні цих кроків в гіршому випадку. За необхідності, знаходження значення тестової точності на тестовій вибірці d_{test}.

Співвідношення розділення між тренувальною та валідаційною вибірками зазвичай обирається в діапазоні 70-80% в залежності від кількості даних, в даному дослідженні було обрано показник в 75% для тренувальної

вибірки. Крім того, за наявності тестової вибірки розділення зазвичай відбувається у відношенні 60% до двох вибірок по 20%. Розділення початкового набору даних на вибірки виконується випадковим чином задля уникнення створення будь-яких закономірностей та узагальнення моделі. Крім того, в усіх вибірках має зберігатися відношення між кількістю даних кожного класу для рівномірного представлення, тож розділення проводиться за допомогою стратифікованого розділення (англ. stratified sampling), тобто окремо для кожного класу.

Таким чином, обраним критерієм оцінки оптимальності роботи моделей є точність. Отже, оптимальною буде вважатися та модель, яка матиме більшу точність або для всіх випадків, або в більшості з них. Це і буде шуканим критерієм знаходження оптимальної моделі.

Аналогічно формується і критерій знаходження оптимального методу згладжування: той метод, застосування якого до обраної перевірочної вибірки буде давати найбільшу точність визначення для заданих наборів даних, буде оптимальним методом згладжування.

3.4. Принципи обрання моделей для поставленої задачі

Існує тенденція до розвитку більш глибоких та складних мереж, хоча ці мережі не завжди достатньо ефективні для класифікації простіших даних в сенсі швидкості роботи та використання можливостей мережі.

Тому в останній час було розроблено декілька прототипів мереж, які мають відносно невелику кількість рівнів, але при цьому виконують обчислення швидко, з ефективним використанням обчислювальних ресурсів та пам’яті, а також з порівняною точністю мереж. Розвиток таких моделей напряму пов’язаний з розвитком мобільних додатків з використанням машинного навчання, безпілотних автомобілів, роботизованої техніки тощо, що вимагає створення моделей нейронних мереж, ефективних з точки зору

часу та ресурсу обчислень на етапі прогнозування значень для вхідних даних, зокрема, для вхідного зображення.

Розпізнавання дорожніх об’єктів відіграє ключову роль в розвитку безпілотних автомобілів та загалом прикладної області, пов’язаної із застосуванням машинного навчання для дорожньої тематики, і швидкість прийняття рішень на основі вирішення задачі класифікації має надзвичайно високе значення, тож застосування згорткових нейронних мереж з великою кількістю рівнів, які вимагають значної кількості математичних операцій для визначення результату для такої сфери часто є занадто неефективним та навіть непідходящим. Натомість, згорткові нейронні мережі з ефективною побудовою рівнів та меншою кількістю параметрів мають значні перспективи в цій галузі. [36] Таким чином, при виборі напряму розробки архітектур для поставленої задачі було вирішено зробити акцент на розгляді тих моделей, які пропонують ефективні моделі рівнів з меншою кількістю параметрів, при цьому зменшуючи складність моделі та підвищуючи її ефективність.

3.5. Обрані моделі згорткових нейронних мереж

Було обрано для розробки дві моделі з класу моделей, вказаних у розділі 3.4. Структури та принципи побудови для обох моделей було оприлюднено в нещодавніх дослідженнях, які вийшли у 2016-2017 роках.

3.5.1. Модель на основі SqueezeNet

Тип згорткової нейронної мережі під назвою SqueezeNet був запропонований у 2016 році дослідниками з Університету Каліфорнії (Берклі), Стенфордського університету та компанії DeepScale як альтернатива згортковим мережам з великою кількістю рівнів, які показують схожі точності на однакових вибірках даних. У даний час існують згорткові

нейронні мережі, які мають надзвичайно велику кількість рівнів та надзвичайно високу точність, проте вимагають величезних ресурсів для навчання та великого об’єму пам’яті для їх використання.

Через це виникає такий парадокс, що для більшості наборів даних існує кілька таких мереж, які можуть досягати великої точності при навчанні, проте будуть неефективними з точки зору практичного використання при відсутності значних обчислювальних можливостей. Таким чином, для заданої точності мережі майже точно SqueezeNet будуть існувати мережі з великою кількістю шарів з тією ж точністю, проте менш ефективні, що є однією з головних переваг мережі. Серед інших переваг цієї мережі автори наводять те, що вона зменшує кількість інформації, що передається між серверами під час паралельного навчання на кількох серверах, її простіше навчати на машинах з меншим об’ємом пам’яті, а також завантажувати до безпілотних автомобілів при оновленні програмного забезпечення. [39]

Головним структурним елементом SqueezeNet є Fire Module, з варіантів якого будується вся система. Він формується наступним чином: спершу він включає шар squeeze (або, якщо перекласти, шар стиснення), що складається з фільтрів розмірності 1x1 та використовується для зменшення кількості вхідних каналів для фільтрів розмірності 3x3. Загалом, одним із принципів побудови SqueezeNet є широка заміна фільтрів розмірностей 3x3 на фільтри з розмірностями 1x1 через набагато меншу кількість параметрів в останніх. Після цього шару його вихідні дані передаються до шару expande (або шар розширення), що складається зі згорток двох розмірностей: 1x1 та 3x3.

Завдяки такій архітектурі, кожний Fire Module має три гіперпараметра: один з них відповідає кількості фільтрів у шарі стиснення та позначається s_{1x1}, два інших відповідають за кількості фільтрів 1x1 та 3x3 відповідно у шарі розширення та позначаються e_{1x1}, e_{3x3}, при цьому перший має бути меншим за суму других для того, щоб кількість вхідних каналів для фільтрів

3x3 в шарі розширення зменшувалася. Структурна схема Fire Module зображена на рис. 3.1.

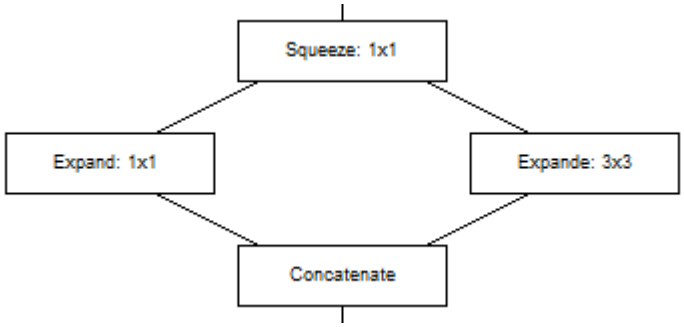


Рисунок 3.1 – Структурна схема Fire Module

Загальна структура SqueezeNet формується таким чином: після вхідного шару розташовується згортковий, після нього йдуть три Fire Module з наступним агрегувальним максимізуючим рівнем, після чого ще чотири Fire Module з агрегувальним максимізуючим рівнем. Після цих шарів розташовується ще один Fire Module з наступними dropout та згортковим та шарами, після чого відбувається глобальне усереднювальне агрегування і зрештою дані передаються до вихідного шару. Слід зазначити, що однією з характерних особливостей моделі є відсутність повнозв’язних шарів, а також розмір кроку для агрегувальних рівнів дорівнює 2.

В реалізованій версії SqueezeNet була запропонована така наступна структура: агрегувальні максимізуючі шари були розташовані після другого та четвертого блоків Fire Module замість третього та сьомого для реалізації однакової кількості блоків з однаковим характером збільшення гіперпараметрів між агрегувальними рівнями, а розміри фільтрів для першого згорткового рівня змінені в сторону збільшення відносно пропонованих 3x3 на встановлені як 7x7 задля кращого масштабування великих об’єктів на зображенні. Вхідна розмірність зображення таким чином збільшена до 227x227, що завдяки використанню згорток 7x7 на першому рівні не складе впливу на ефективність мережі, проте покращить розміщення ознак на вхідному зображенні.

SqueezeNet має щонайменше в кілька десятків разів параметрів менше, ніж згорткові мережі зі схожою точністю, але більшою кількістю рівнів, що відповідно зменшує і розмір мережі як такої.

Повна схема реалізованої моделі мережі SqueezeNet наведена на рис. 3.2.

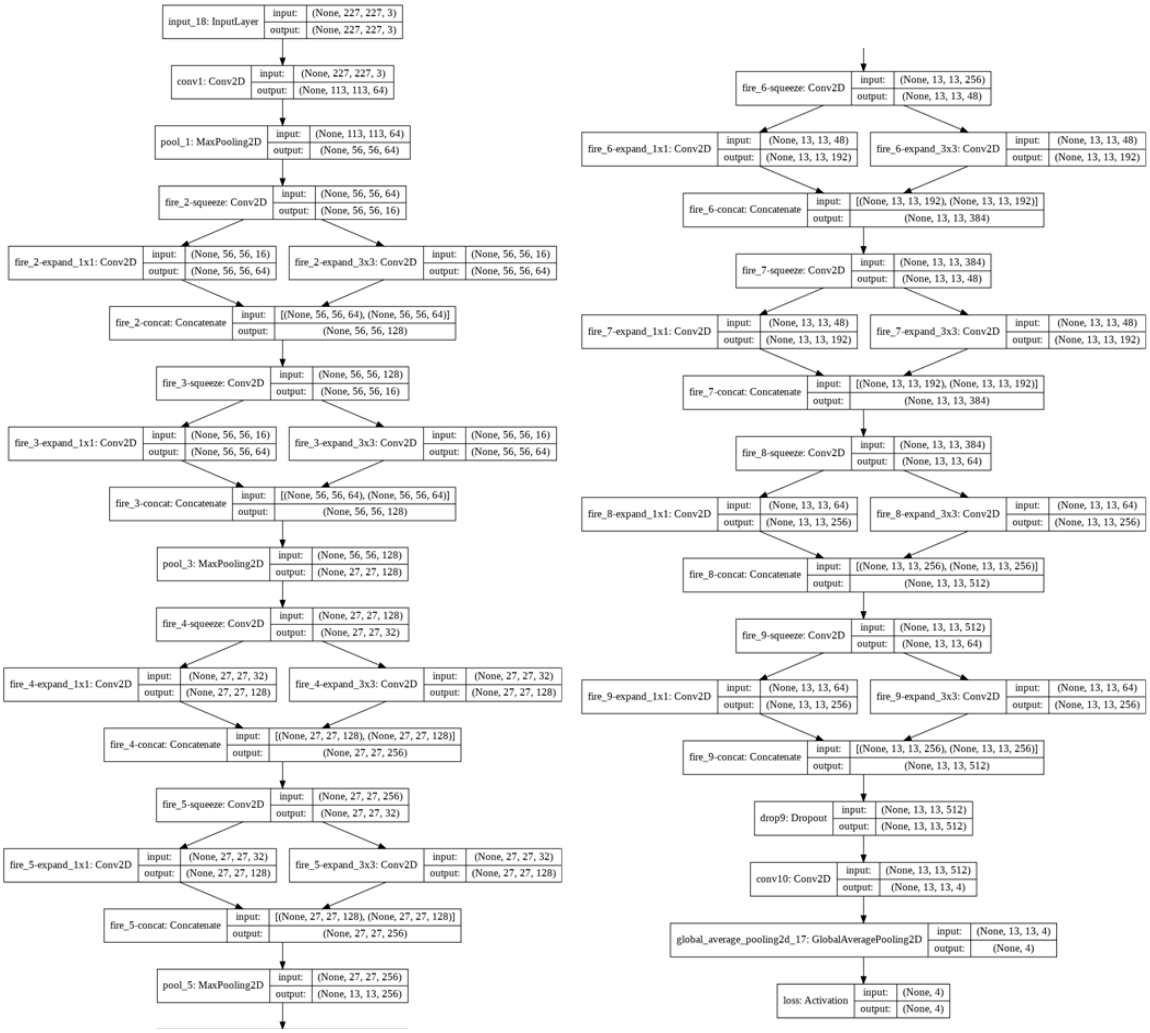


Рисунок 3.2 – Повна структурна схема реалізованої моделі мережі SqueezeNet

3.5.2. Модель на основі MobileNet

Тип згорткової нейронної мережі MobileNet був запроваджений у 2017 році дослідниками з компанії Google. [37] Цей клас мереж використовує поглиблені розділені згортки (англ. depthwise separable convolutions), за допомогою яких досягає зменшення кількості параметрів мережі. Вперше

цей тип згортки в сучасному вигляді з’явився у мережі Xception, яка була запропонована Ф. Шалеттом у 2016 році.

Такий тип згортки поєднує в собі дві операції, а саме поглиблену згортку (англ. depthwise convolution) та поточкову згортку (англ. pointwise convolution). [38] На відміну від звичайних згорток, де кожен фільтр діє на всі канали вхідного зображення, в поглиблених згортках на кожний вхідний канал припадає окрема згортка, тобто фільтр з глибиною 1. Це забезпечує однакову кількість каналів на вході та на виході поглибленої згортки. Вихід цієї операції використовується для поточкової згортки, яка є власне згорткою розмірності 1x1, яка використовується для обробки результатів поглибленої згортки.

Як результат заміни звичайних згорток на поглиблені розділені, кількість параметрів на окремому рівні зменшується в кілька разів без втрати характеристик відповідних операцій. Крім того, в мережі MobileNet застосовується ще один спосіб зменшення мережі, який полягає у введенні двох додаткових гіперпараметрів: множника ширини (англ. width multiplier) та множника розмірності (англ. resolution multiplier). Функція множника ширини $\alpha \in [0; 1]$ полягає в скороченні кількості каналів на кожному рівні: при його застосуванні всі кількості вхідних та вихідних каналів на кожному рівні множаться на це число, а їх загальна кількість зменшується приблизно на α^2 від неї.

Аналогічно, множник розмірності $\rho \in [0; 1]$ зменшує розмірність вхідного зображення в стільки ж разів, хоча таке зменшення можна зробити і шляхом зменшення вхідної розмірності, зважаючи на наявність глобального усереднювального агрегувального рівня у мережі. Значення обидвох множників за замовчуванням дорівнюють 1.

Загальна структура мережі нараховує 30 рівнів, серед яких згорткові звичайні, поглиблені та поточкові рівні. Така структура моделі забезпечує швидку роботу мережі у порівнянні з іншими мережами зі схожою

результуючою точністю. Структура занадто велика для розміщення на одному малюнку, тому на рис. 3.3. зображено перші 11 рівнів з першими звичайною, поглибленою та поточковою згорткою.

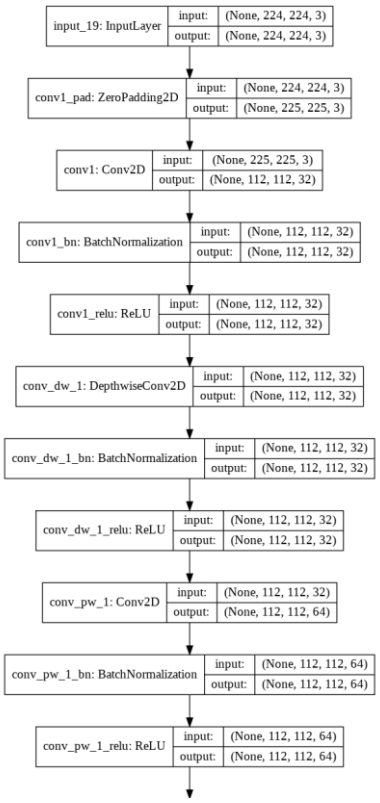


Рисунок 3.3 – Перші 11 рівнів структури мережі MobileNet

3.6. Принципи застосування методів зменшення рівня шуму до моделей

Для отримання та порівняння результатів роботи моделей та методів зменшення рівня шуму з метою виявлення оптимальних рішень необхідно встановити єдині правила застосування методів зниження шуму в зображення при роботі моделей. Для вибраних наборів даних, що детально розглянуто в двох наступних розділах, виконується навчання обраних моделей на тренувальній вибірці та перевірка точності їх роботи за допомогою валідаційної вибірки. Після завершення навчання моделі її робота має бути перевірена на зашумленій валідаційній вибірці, в результаті чого

має бути зроблений висновок про вплив шуму в зображеннях на роботу обраних моделей.

Для оцінки роботи та обробки даних для зашумлених зображень можливі два принципово відмінні варіанти:

1) Зашумлені дані розпізнаються тією ж самою моделлю, що й чисті, проте до зображень при виявленні на них шуму спершу застосується певний алгоритм зменшення його рівня, а вже після цього відновлене зображення обробляється та класифікується мережею. Такий випадок є можливим завдяки наявності методів оцінки рівня шуму, які дозволяють розпізнати, чи необхідне для даного зображення його обробка методом зниження рівня шуму. Критерій для такого
2) Відбувається навчання додаткової мережі з тією ж моделлю, призначення якої полягатиме саме в розпізнаванні зашумлених зображень, які попередньо були відновлені. Така мережа має навчатися на даних, які є відновленими зашумленими зображеннями, та використовуватися тільки для них.

Створення та підтримка мережі, зазначеної у варіанті 2, вимагатиме додаткових витрат пам’яті та ресурсів системи, крім того, під час розпізнавання зображень така додаткова мережа має або знаходитися в пам’яті одночасно з основною для миттєвого перекладення задачі розпізнавання у випадку виявлення зашумленого зображення, або постійно туди підвантажуватися в таких ситуаціях, що вимагатиме ще більших витрат ресурсів та, що найважливіше, витрат часу. Отже, з запровадженням такої мережі може бути знівельовано ефект ефективної побудови моделі згорткової нейронної мережі, оскільки її розмір фактично подвоюється. Крім того, така мережа в загальному та на нижчих рівнях буде навчатися приблизно однаково до основної, бо за однакової структури та однакових класів для

розпізнавання вплив від алгоритмів зменшення рівня шуму може бути малим для загальних ознак зображення, як кутів тощо.

Тому, зважаючи на ці фактори, було обрано до реалізації варіант 1, в якому для розпізнавання спершу зашумлених, а пізніше відновлених даних буде використовуватися та ж сама модель, що і для чистих даних. В такому варіанті застосування методів знешумлення даних відбуватиметься або ще до їх подання до безпосередньої програми розпізнавання та класифікації, тобто всі зображення для мережі будуть однаковими, або ж безпосередньо обробленням даних після їх отримання з зображення, але перед поданням до нейронної мережі. Тому, виходячи з цього, було реалізовано два механізми обробки зашумлених зображень перед їх розпізнаванням та класифікацією: в першому модель розпізнає вхідне зображення без урахування його рівня шуму, в такому випадку необхідно, щоб зображення було попередньо за необхідності оброблене алгоритмом зменшення рівня шуму; в другому модель при обробці зображення проводить оцінку рівня шуму в ньому, за необхідності проводить зменшення його рівня за алгоритмом та використовує нову тимчасову знешумлену версію зображення як вхідне для задачі розпізнавання. Другий алгоритм дозволяє системі працювати з зображеннями в реальному часі, перший алгоритм дозволяє задавати бажані налаштування для обробки зображень перед їх розпізнаванням.

3.7. Результати роботи для набору даних Traffic Signs Dataset

Для набору даних Traffic Signs Dataset, детальний опис якого було наведено у розділі 2.4.1, було реалізовано обидві з запропонованих моделей згорткових нейронних мереж. Після цього за схемою, наведеною у розділі 3.3, було проведено навчання для кожної з мереж, яке проводилося до того моменти, коли з’явилися явні ознаки перенавчання і валідаційна точність для вибірки припинила підвищуватися.

Для обох моделей було застосовано алгоритм Adam для навчання, оскільки він виявив найкращі результати при навчанні мережі, а також уникав перенавчання, яке виникало для алгоритму стохастичного градієнта вже після кількох кроків. Була застосована технологія запам’ятовування найкращої точки, коли версія моделі з найкращою метрикою (валідаційною точністю) зберігається та представляється як остаточна версія моделі. Завдяки цьому є можливим зберегти результати навчання навіть у випадку, коли модель почала перенавчатися.

Загалом для навчання моделі на основі SqueezeNet знадобилося 150 епох (ітерацій оновлення параметрів), тоді як для навчання моделі на основі MobileNet знадобилося 120 епох до того моменту, коли перенавчання стало помітним фактором та покращення валідаційної точності зупинилося.

Перед початком навчання описаний набір даних було за допомогою спеціально написаної програми випадковим чином стратифіковано розділено на тренувальну, валідаційну та тестову вибірки. Розділення було виконано у відношенні 60%:20%:20% відповідно до вимог, визначених у розділі 3.3. Загальний об’єм для отриманих вибірок склав 1313 зображень в тренувальній вибірці, 436 у валідаційній та 439 у тестовій. Дещо різна кількість зображень у валідаційній та тестовій вибірці пояснюється тим, що кількість зображень у наборі даних не кратна 100%.

За результатами роботи запропонованих моделей згорткових нейронних мереж та використання методів згладжування шуму було отримано наступні результати:

Таблиця 3.1 – результати роботи моделей та методів знешумлення для набору даних Traffic Signs Dataset, тестова точність

Модель	Ориг. набір	Зашумлені дані	ВПВ	ВПV	МПВб	МПВд	ДФ
SqueezeNet	0,8269	0,5490	0,5421	0,5444	0,6720	0,7608	0,7130
MobileNet	0,7350	0,6036	0,6014	0,6082	0,6492	0,6970	0,6720

Де ВПВ – вайвлет-перетворення з використанням методу BayesShrink, ВПVВ – вайвлет-перетворення з використанням методу VisuShrink, МПВб – метод повної варіації з оптимізацією Брегмана, МПВд – метод повної варіації з багатовимірними зображеннями, ДФ – двосторонній фільтр.

3.8. Результати роботи для набору даних Graz02

Для набору даних Graz02, детальний опис якого наведено у розділі 2.4.2, аналогічно до набору даних Traffic Signs Dataset, було реалізовано обидві з запропонованих моделей. Було проведено процес навчання за схемою, зазначеною в розділі 3.3. В якості алгоритму для навчання також було обрано алгоритм Adam для навчання, оскільки мережа навчалася швидше та якісніше при його використанні, а також дозволив виконати навчання на більш ніж 80 епохах для кожної з моделей до того моменту, коли почали з’являтися ознаки перенавчання. Для збереження найкращих результатів навчання була знову застосована технологія запам’ятовування найкращої точки зі збереженням версії моделі з найкращим значенням встановленої метрики (валідаційної точності).

Для навчання моделі на основі SqueezeNet було пройдено 130 епох оновлення параметрів до появи перенавчання, тоді як для навчання моделі на основі MobileNet знадобилося 90 епох. Проте, як продемонстровано нижче в

результатах роботи, швидкість навчання не мала вирішального значення у визначенні якості роботи моделі.

Перед початком навчання набір даних було випадковим чином було стратифіковано розділено на тренувальну та валідаційну вибірки. Загальний об’єм для отриманих вибірок склав 1108 зображень для тренувальної та 368 зображень для валідаційної.

За результатами роботи запропонованих моделей та використання методів згладжування шуму було отримано наступні зведені результати:

Таблиця 3.2 – результати роботи моделей та методів знешумлення для набору даних Graz02, валідаційна точність

Модель	Ориг. набір	Зашу-млені дані	ВПВ	ВПV	МПВ б	МПВ д	ДФ
SqueezeNet	0,9728	0,9348	0,9511	0,9538	0,9592	0,9647	0,9457
MobileNet	0,875	0,8179	0,8668	0,8668	0,875	0,8697	0,8697

3.9. Аналіз результатів для згладжування зображень

Як бачимо, виходячи з отриманих результатів для заданих наборів даних, результати для моделі на основі SqueezeNet виявилися загалом набагато точнішими за аналогічні результати для моделі на основі MobileNet, окрім випадку з зашумленими даними для набору даних Traffic Signs Dataset, де точність для моделі на основі SqueezeNet опустилася нижче для відповідної точності для моделі на основі MobileNet, що збереглося також і для обох варіантів методу зниження рівня шуму за допомогою вейвлет-перетворення. Так, для всіх ситуацій, окрім згаданих, точність для двох мереж відрізнялася, і в деяких випадках значно, і $a_i^{SN} > a_i^{MN} \; \forall i$, де a_i^{SN} –

точність для моделі SqueezeNet в i-му випадку, a_i^{MN} – аналогічне значення для моделі MobileNet.

Це свідчить про нижчу стійкість першої моделі для шуму, але цей недолік повністю зникає при застосуванні методів зниження шуму з використанням повної варіації або двостороннього фільтру, що робить модель SqueezeNet оптимальним вибором. Ця модель продемонструвала найвище значення точності в обох випадках, яка у випадку для набору даних Traffic Signs Dataset склала 82,69 %, а для набору даних Graz02 97,28 %. Тож, заважаючи на отримані результати та встановлені критерії, можемо з упевненістю сказати, що модель SqueezeNet буде оптимальним вибором для задачі розпізнавання дорожніх об’єктів в цих умовах.

З результатами для методів зниження рівня шуму не все так однозначно. Метод згладжування за допомогою вайвлет-перетворення не зайняв два останніх місця тільки для моделі на основі SqueezeNet для набору даних Graz02, де найгіршим став метод з використанням двостороннього фільтру, проте в цьому випадку він все ще покращує результати у порівнянні з зашумленою частиною, але для набору даних Traffic Signs Dataset він не тільки не впливає на результати роботи моделей у порівнянні із зашумленими даними, а ще й незначним чином погіршує її, ставши єдиним таким алгоритмом. Тобто, цей метод можна точно не розглядати як оптимальний, проте його можна вважати найгіршим методом для застосування в такій задачі.

Метод згладжування з двостороннім фільтром непогано показав себе для набору даних Traffic Signs Dataset, а також в моделі MobileNet для Graz02, проте для моделі SqueezeNet та того ж набору даних він став передостаннім. Вказані властивості та його здатність до значного поліпшення результатів роботи моделей, в одному з випадків навіть на більш ніж 16 %,

дозволяють вважати його можливим та корисним для використання в таких задачах, проте через відсутність найкращих показників не оптимальним.

Серед двох варіантів, що залишаються, є два варіанти методу повної варіації, які в усіх випадках входили до трійки найкращих методів, при цьому ітеративний метод став найкращим три рази, а метод з оптимізацією Брегмана – одного. Слід зазначити, що однозначно визначити серед них оптимальний, виходячи з результатів для набору даних Graz02 важко, адже, наприклад, загальне абсолютне відхилення в цьому випадку для МПВб склало 0,136, а для МПВд – 0,134, кількість неправильно визначених зображень для них однакова, тож для цього набору оптимальним методом знешумлення може вважатися метод повної варіації з використанням одного з двох способів його обчислення. З іншого боку, результати для набору даних Traffic Signs Dataset показують, що перевага на боці ітеративного методу не випадкова та спостерігається на цьому наборі, фактично ставши найкращим алгоритмом наближення, в одному з випадків випередивши другий кращий метод з двостороннім фільтром на 4,78 %. Тож перевагу за точністю та з точки зору того, що модель на основі SqueezeNet визнано оптимальною, має ітеративний метод повної варіації.

Слід зазначити, що цифровий шум вказав значний вплив на точність роботи моделей, зменшивши валідаційну точність на більш ніж 10% для набору даних Traffic Signs Dataset та на приблизно 4-6% для набору Graz02 для кожної з моделей. Хоча результати для другого набору виглядають менш значними, ніж для першого, проте необхідно вказати на високе початкове значення точності для нього, а також той факт, що частка неправильно визначених даних для моделі SqueezeNet збільшилася в 2,4 рази, а для моделі MobileNet в 1,45 разів. Це свідчить про більшу стійкість до зашумленості моделі на основі SqueezeNet, оскільки в неї точність зменшилася на менше значення і яка до того ж мала більшу точність і на чистій вибірці, хоча у

відносних значеннях вона почала робити більше помилок у порівнянні з чистими даними.

3.10. Висновки до розділу 3

В цьому розділі було розглянуто та реалізовано моделі згорткових нейронних мереж для вирішення задачі розпізнавання дорожніх об’єктів, які було визначено за попередньо сформульованими принципами обрання моделей для вирішення такої задачі, а саме було обрано моделі на основі мереж SqueezeNet та MobileNet. Запроваджено критерії знаходження оптимальної моделі та методу згладжування, при цьому розглянуто метрики оцінювання ефективності роботи мереж та розбиття набору даних на тренувальну та валідаційну вибірку, сформовано алгоритм дій при навчанні мережі та визначено критерій оптимальної моделі як модель з найменшим значенням валідаційної точності. Також було розглянуто вимоги до техніки та програмного забезпечення для використання розробленого програмного продукту, а також використані для його розробки засоби мови програмування

Вказані методи та критерії було застосовано до обраних наборів даних, внаслідок чого було проведене тренування застосованих методів та в результаті отримано більше 10 різних моделей для різних ситуацій, кожна з яких характеризується своєю валідаційною точністю. Отримані результати занесено до таблиць та проведено за їх результатами аналіз, в ході якого виявлено, що оптимальним методом вирішення задачі буде модель на основі SqueezeNet, а методом зниження рівня шуму – ітеративний метод повної варіації для багатовимірних зображень.

РОЗДІЛ 4
ФУКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1. Вступ

В цьому розділі проводиться оцінка основних вагомих складових частин програмного продукту з побудови моделі згорткової нейронної мережі та застосування до задачі розпізнавання та класифікації зображень. Програмний продукт було створено за допомогою мови програмування Python в середовищі Jupyter Notebook з використанням хмарної системи Google Colaboratory для навчання моделей.

Для цих задач було використано апарат функціонально-вартісного аналізу (ФВА). Функціонально-вартісний аналіз є технологією, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам незалежно від потрібних на кожному етапі виробництва обсягів ресурсів. Дії, виконані на цих етапах, у контексті методу ФВА називають функціями.

Суть методу ФВА полягає у забезпеченні правильного розподілу на прямі та непрямі витрати ресурсів, які були виділені на виробництво продукції чи надання послуг. В цьому випадку метод полягає в аналізі функцій програмного продукту та виявленні усіх витрат на реалізацію цих функцій.

4.2. Постановка задачі проектування

Проводиться оцінка основних характеристик програмного продукту, призначеного для створення, визначення та навчання згорткових нейронних

мереж та зниження рівня шуму в зображеннях, що використовуються мережею. Інтерфейс користувача було розроблено за допомогою мови програмування Python в середовищі розробки Jupyter Notebook та хмарній системі Google Colaboratory з використанням бібліотек skimage, matplotlib мови Python.

В розділах нижче наведено аналіз різних варіантів реалізації продукту з метою вибору оптимального, з огляду як на економічні фактори, так і на характеристики продукту, які впливають на продуктивність його роботи та сумісність з технічним та апаратним забезпеченням, так, побудований продукт має забезпечувати можливість обробки великих об’ємів даних в реальному часі, працювати на машинах з об’ємом ресурсів, не більшим за стандартний, зручність при роботі з користувачем та зрозумілість його інструментів.

4.3. Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір класу моделі для задачі розпізнавання та класифікації;

F_3 – відображення візуальних результатів роботи.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Python;

б) мова програмування C++;

Функція F_2:

а) моделі з оптимізованою структурою параметрів;

б) моделі з надвеликою кількістю рівнів.

Функція F_3:

а) візуалізація результатів за допомогою бібліотек skimage та matplotlib;

б) інтерфейс користувача з візуалізацією, створений за допомогою технології Qt.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

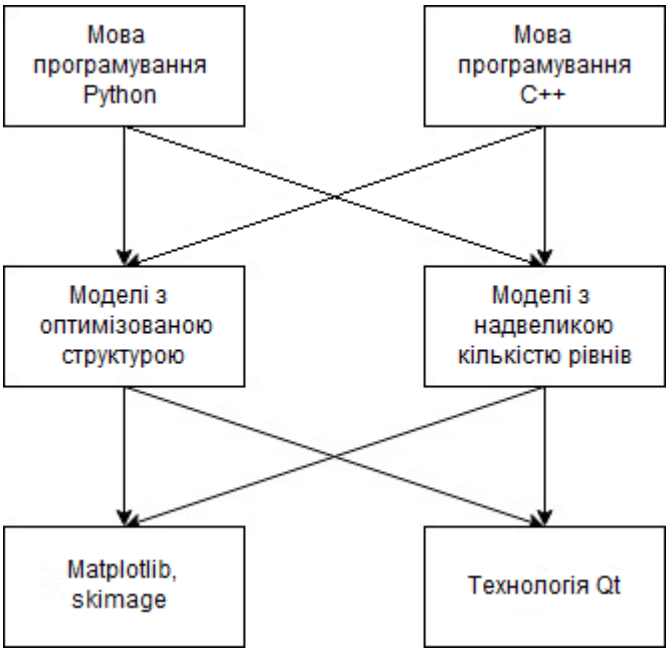


Рис. 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП. На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

Таблиця 4.1. Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Менший об’єм коду, менше часу на написання програми, більше вбудованих бібліотек	Ускладнена пряма робота з пам’яттю, складніше формування проекту
	Б	Легко вбудовується в проект, кросплатформеність	Вимагає велику кількість часу для написання
F2	*А*	Малий обсяг необхідних ресурсів, швидке навчання	Можливі складнощі при виявленні великих ознак
	Б	Виявляються всі ознаки зображень	Надзвичайно високі вимоги до необхідних ресурсів та часу
F3	*А*	Гнучкість у роботі, простота створення	Відсутня можливість роботи з вікнами
	Б	Можливість створення інтерфейсу у вікні	Вимагає більше ресурсів, нижча швидкість

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Враховуючи специфіку поставленої задачі, для якої має проводитися робота з великими об’ємами даних, а також значним розміром моделей, перевагою є наявність стандартних засобів роботи з ними, які окрім того можуть покращувати швидкодію програми, а також легкість у реалізації моделей. Таким чином, варіант б) має бути відкинутий.

Функція *F2:*

Оскільки розпізнавання та класифікація дорожніх об’єктів вимагає високу швидкість роботи продукту та можливість його запуску на пристроях

з обмеженим об’ємом ресурсів, для яких розмір пам’яті не може перевищувати декількох Гб, то пріоритетною є ефективність роботи моделі за необхідної точності, тому варіант б) має бути відкинутий

Функція *F3:*

Інтерфейс користувача не відіграє велику роль для відображення кінцевих результатів продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

F1а – F2а – F3а

F1а – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.4. Обґрунтування системи параметрів ПП

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X1 – швидкодія мови програмування;

X2 – об’єм пам’яті для збереження даних;

X3 – час обробки даних;

X4 – час навчання для даних;

X5 – потенційний об’єм програмного коду.

X1: Відображає швидкість роботи операцій обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії, необхідні для обробки даних.

X4: Відображає час, який витрачається на навчання моделей для отримання результатів для даних.

X5: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2. Основні параметри ПП

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	11000	14000	19000
Об’єм пам’яті для збереження даних	X2	Мб	1500	750	500
Час обробки даних	X3	мс	1200	690	180
Час навчання для даних	X4	год	12	6	2
Потенційний об’єм програмного коду	X5	кількість рядків коду	5000	3500	1500

За даними таблиці 2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.6.

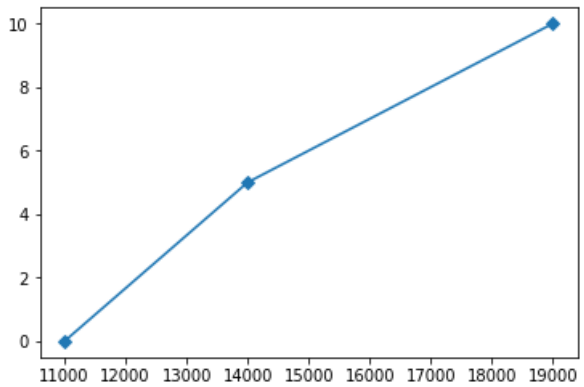


Рисунок 4.2 – X1, швидкодія мови програмування

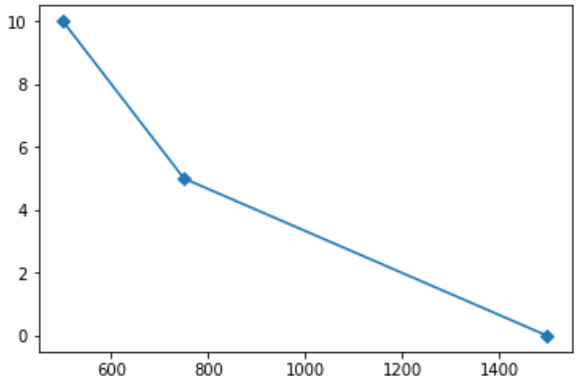


Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

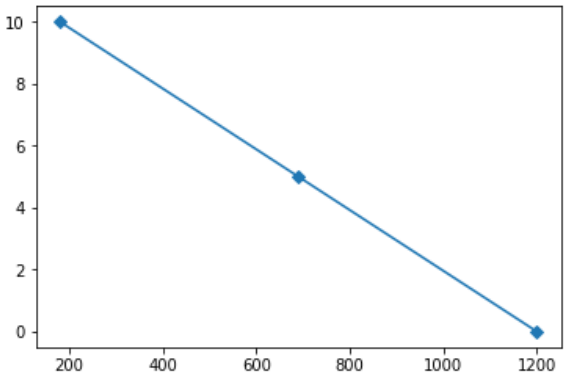


Рисунок 4.4 – X3, час обробки даних

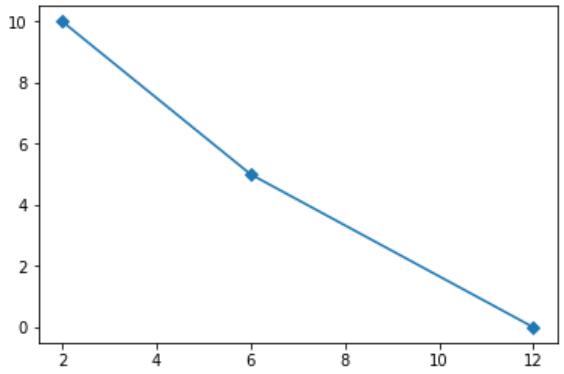


Рисунок 4.5 – X4, час навчання для даних

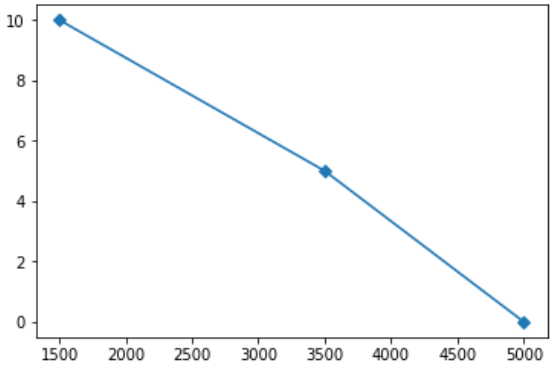


Рисунок 4.6 – X5, потенційний об’єм програмного коду

4.5. Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	оп/мс	2	1	2	1	1	2	1	10	-11	121
X2	Об’єм пам’яті для збереження даних	Мб	4	5	3	5	2	4	4	27	6	36
X3	Час обробки даних	мс	1	2	1	2	3	1	2	12	-9	81
X4	Час навчання для даних	год	3	4	4	3	5	3	3	25	4	16
X5	Потенційний об’єм програмного коду	кількість рядків коду	5	3	5	4	4	5	5	31	10	100
	Разом		15	15	15	15	15	15	15	105	0	354

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 21;$$

в) відхилення суми рангів кожного параметра від середньої суми

рангів:

$$\Delta_i = R_i - T.$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 354.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 354}{7^2(5^3 - 5)} = 0.7224 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1.5
X1 і X3	<	>	<	>	>	<	>	>	1.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X1 і X5	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	>	<	<	<	0.5
X2 і X4	<	<	>	<	>	<	<	<	0.5
X2 і X5	>	<	>	<	>	>	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X3 і X5	>	>	>	>	>	>	>	>	1.5
X4 і X5	>	<	>	>	<	>	>	>	1.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j. \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{N} b_i},$$

де $b_i = \sum_{i=1}^{N} a_i.$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{N} b_i'},$$

де $b_i' = \sum_{i=1}^{N} a_{ij} b_j.$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – розрахунок вагомості параметрів

Параметри x_i	Параметри x_j					Перша ітерація		Друга ітерація		Третя ітерація	
	1	2	3	4	5	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	1,5	1,5	1,5	1,5	7	0.28	34	0.296	155.5	0.296
X2	0,5	1,0	0,5	0,5	1,5	4	0.16	17.5	0.152	80.25	0.153
X3	0,5	1,5	1,0	1,5	1,5	6	0.24	27.5	0.239	124.75	0.238
X4	0,5	1,5	0,5	1,0	1,5	5	0.2	22	0.191	100	0.190
X5	0,5	0,5	0,5	0,5	1,0	3	0.12	14	0,122	64.5	0.123
Всього:						25	1	115	1	525	1

4.6. Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів та X1 (швидкодія мови програмування), X2 (об’єм пам’яті для збереження даних) та X4 (час навчання для даних) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3 (час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 360 мс або варіанту б) 800 мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{ві},j} B_{i,j},$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості *і*–го параметра;

B_i – оцінка *і*–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	14000	5	0.296	1.48
F2	А	750	5	0.153	0.765
F3	А	360	8,2	0.238	1.952
	Б	800	3,9	0.190	0.741

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \cdots + K_{\text{ТУ}}[F_{zk}]$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1{,}48 + 0{,}765 + 1{,}952 = 4{,}197;$$
$$K_{K2} = 1{,}48 + 0{,}765 + 0{,}741 = 2{,}986.$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.7. Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки та моделей мереж.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М},$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$T2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

TI = (122.4 + 19.44 + 4.8 + 19.44) · 8 = 1328,64 людино-годин.

TII = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин.

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь програміст з окладом 14000 грн., один аналітик в області даних з окладом 20000 грн, та один інженер даних з окладом 15000 грн. Визначимо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн.,}$$

де M – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{14000 + 20000 + 15000}{3 \cdot 21 \cdot 8} = 97{,}22 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{ЗП} = C_{ч} \cdot T_i \cdot K_{Д},$$

де $C_{ч}$– величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$K_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $C_{ЗП} = 97{,}22 \cdot 1328.64 \cdot 1.2 = 155004.46$ грн.

II. $C_{ЗП} = 97.22 \cdot 1345.52 \cdot 1.2 = 156973.75$ грн.

Відрахування на соціальний внесок становить 22%:

I. $C_{ВІД} = C_{ЗП} \cdot 0.22 = 155004.46 \cdot 0.22 = 34100.98$ грн.

II. $C_{ВІД} = C_{ЗП} \cdot 0.22 = 156973.75 \cdot 0.22 = 34534.23$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($C_{М}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 14000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{Г} = 12 \cdot М \cdot K_{З} = 12 \cdot 14000 \cdot 0{,}2 = 33600 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = C_{Г} \cdot (1 + K_{З}) = 33600 \cdot (1 + 0.2) = 40320 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 40320 \cdot 0,22 = 8870,4 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 22000 грн.:

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 22000 = 6325 \text{ грн.,}$$

Де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 22000 \cdot 0.05 = 1265 \text{ грн.,}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.9 = 1677.6 \text{ годин,}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677.6 \cdot 0,56 \cdot 0,2 \cdot 2,7515 = 516.98 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 22000 \cdot 0,67 = 14740 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н,$$

$$С_{ЕКС} = 40320 + 8870,4 + 6325 + 1265 + 516.98 + 14740 = 72037.38 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 72037.38/1677.6 = 42.94 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т,$$

I. $С_М = 42.94 \cdot 1328,64 = 57051.80$ грн.

II. $С_М = 42.94 \cdot 1345.52 = 57776.63$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67,$$

I. $С_Н = 155004.46 \cdot 0,67 = 103852.99$ грн.

II. $С_Н = 156973.75 \cdot 0,67 = 105172.41$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н,$$

I. $С_{ПП} = 155004.46 + 34100.98 + 57051.80 + 103852.99 = 350010.23$ грн.

II. $С_{ПП} = 156973.75 + 34534.23 + 57776.63 + 105172.41 = 354456.61$ грн.

4.8. Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$$К_{ТЕР1} = 4.197 / 350010.23 = 1,199 \cdot 10^{-5},$$

$$К_{ТЕР2} = 2.986 / 354456.61 = 8,424 \cdot 10^{-6}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1,199 \cdot 10^{-5}$.

4.9. Висновки до розділу 4

В цьому розділі було проведено повний функціонально-вартісний аналіз програмного продукту, розробленого в дипломній роботі. При цьому в ході аналізу продукт спершу було досліджено з технічної точки зору, так, визначено основні його функції та сформовано множину варіантів їх реалізації, а також обчислено коефіцієнт технічного рівня для визначення оптимальної з технічної точки зору реалізації функцій продукту. Після цього було проведено дослідження економічної обґрунтованості альтернатив реалізацій за допомогою коефіцієнта ефективності, для знаходження якого були обчислені допоміжні параметри, як трудомісткість, витрати на заробітну плату, вартість машино-години.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,44 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- використання моделей з оптимізованою структурою параметрів;
- візуалізація результатів за допомогою бібліотек skimage та matplotlib.

Даний варіант виконання програмного комплексу дає користувачу швидку роботу програми, зручний та гнучкий інтерфейс, а також гарний функціонал зі швидкими строками його реалізації.

ВИСНОВКИ

В дипломній роботі було виконано дослідження, присвячене застосуванню згорткових нейронних мереж до розпізнавання дорожніх об’єктів в умовах зашумленості.

Було проведено огляд згорткових нейронних мереж, їх характеристик та методів, та розглянуто варіанти їх реалізації та особливості їх застосування до розпізнавання та класифікації зображень, а також необхідні для цього інструменти.

Було розглянуто поняття цифрового шуму зображення, причини його утворення та проведене дослідження щодо найчастіших його видів, закономірностей в ньому, а також способів його усунення за допомогою методів зменшення рівня шуму з використанням математичного апарату. Внаслідок цього було обрано три методи згладжування зображення (два з яких мають по два варіанти реалізації), для яких було реалізовано програмний продукт.

Було досліджено оптимальність використання згорткових нейронних мереж до поставленої задачі на прикладі двох наборів реальних даних, які включають в себе різні види дорожніх знаків, велосипеди, автомобілі тощо. Конкретні методи для реалізації згорткових нейронних мереж було обрано з урахуванням того, що дана задача здобуває широке поширення в системах, що працюють в режимі реального часу та коректність їх роботи залежить від ефективності їх інструментів, тому було обрано методи з оптимізованою структурою операцій всередині нейронної мережі. Моделі мереж було об’єднано з методами зменшення рівня шуму в програмний продукт таким чином, що його застосування можливе як для масиву даних, так і в реальному часі.

Розроблені методи та моделі було застосовано до реальних наборів даних дорожніх об’єктів, де моделі показали високий рівень точності

прогнозування, максимальний знайдений рівень точності для чистих даних склав 97,28 %. Роботу згорткових нейронних мереж було перевірено для різних видів даних та різних методів зменшення рівня шуму, внаслідок чого виявлено, що одна з реалізованих моделей за нормальних умов вирішує задачу краще за другу, проте менш стійка до появи шуму на зображеннях.

Було проаналізовано вплив алгоритмів зменшення рівня шуму на результати роботи мереж та виявлено, що їх застосування в загальному випадку покращує роботу алгоритмів, інколи доводячи рівень точності близько до початкового. Проте, для одного з наборів даних методи з використанням вейвлет-перетворення відчутно не покращували результати, але найоптимальнішим методом зменшення рівня шуму було визначено ітеративний метод повної варіації, хоча методи повної варіації Брегмана та метод з двостороннім фільтром також показали гарні результати.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bengio Y., Lecun, Y. Convolutional Networks for Images, Speech, and Time-Series. 1997. URL: https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series/links/0deec519dfa2325502000000.pdf
2. Bengio Y., Lecun, Y. Convolutional Networks for Images, Speech, and Time-Series. 1997. URL: https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series/links/0deec519dfa2325502000000.pdf
3. Khandelwal R. Convolutional Neural Network (CNN) Simplified. 2018. URL : https://medium.com/datadriveninvestor/convolutional-neural-network-cnn-simplified-ecafd4ee52c5
4. Stockman G. C., Shapiro L. G. Computer Vision. Prentice Hall, February 2001. 609 p.
5. Rao D., McMahan B. Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. O'Reilly Media, 2019. 256 p.
6. Springenberg J. T., Dosovitskiy A., Brox T., Riedmiller M. Striving for Simplicity: The All Convolutional Net. arXiv preprint: 1412.6806. 2014.
7. Scherer, Dominik; Müller, Andreas C.; Behnke, Sven (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. / 20th International Conference on Artificial Neural Networks (ICANN), Thessaloniki, Greece: Springer, 2010. с. 92–101.
8. Graham B. Fractional max-pooling. arXiv preprint: 1412.6071. 2014.
9. Khan S., Rahmani H., Ali Shakh S. A., Bennamoun M. A Guide to Convolutional Neural Networks for Computer Vision. Morgan & Claypool Publishers, 2018. 207 p.
10. Brownlee J. Loss and Loss Functions for Training Deep Learning Neural Networks, 2019. URL : https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/
11. Brownee J. How to Choose Loss Functions When Training Deep Learning Neural Networks. 2019. URL : https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

12. Goodfellow I., Bengio Y., Courville A. Deep Learning (Adaptive Computation and Machine Learning series). The MIT Press, 2016. 775 p.
13. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, Vol. 60 Issue 6, June 2017 : ACM New York, NY, USA. P. 84-90. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
14. Gonzalez R. C., Woods R. E. Digital Image Processing. Prentice Hall, Upper Saddle River, New Jersey, 2008. 954 p.
15. Veererajan T. Probability, Statistics And Random Processes. Tata McGraw-Hill, 2002. 693 p.
16. Simmons J. P., Drummy L. F., Bouman C. A., De Graef M. Statistical Methods for Materials Science: The Data Science of Microstructure Characterization. CRC Press, 2019. 514 p.
17. Bottacchi S. Multi-Gigabit Transmission over Multimode Optical Fibre: Theory and Design Methods for 10GbE Systems. John Wiley & Sons, Chichester, UK, 2006. 670 p.
18. De Wos W. H., Munck S., Timmermans J.-P. Focus on Bio-Image Informatics. Springer, 2016. 272 p.
19. Gelman L. Advances in Electrical Engineering and Computational Science. Springer Science & Business Media, 2009. 726 p.
20. Plötz T., Roth S. Benchmarking Denoising Algorithms with Real Photographs. arXiv preprint: 1707.01313. 2017.
21. Manjon J. V., Coupe P. MRI Denoising Using Dep Learning. / Patch-Based Techniques in Medical Imaging: 4th International Workshop, Patch-MI 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings. Springer, 2018. p. 12-19.
22. Donoho D. L., Johnstone I. M. Ideal Spatial Adaptation by Wavelet Shrinkage. / Biometrika 81.3, 1994. p. 425-455.
23. Tiagi V. Understanding Digital Image Processing. CRC Press, 2018. 368 p.
24. Cohen R. Signal Denoising Using Wavelets. Project Report. – Department of Electrical Engineering Technion, Israel Institute of Technology, Haifa, 2012.
25. Chang C. G., Yu B., Vetterli M. Adaptive Wavelet Thresholding for Image Denoising and Compression. / IEEE Trasactions on image processing, vol. 9, no. 9, 2000. p. 1532-1536

26. Getreuer P. Rudin–Osher–Fatemi Total Variation Denoising using Split Bregman. / Image Processing On Line, 2012.
27. Goldstein T., Osher S. The Split Bregman Method For L1 Regularized Problems. / SIAM Journal on Imaging Sciences 2(2), 2009. p. 323-343.
28. Chambolle A. An algorithm for total variation minimization and applications. / Journal of Mathematical Imaging and Vision. Springer, 2004. p. 89-90.
29. Tomasi C., Manduchi R. Bilateral Filtering for Gray and Color Images. / Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). Bombay, 2002.
30. Paris S., Kornprobst P., Tumblin J., Durand F. Bilateral Filtering: Theory and Applications. Now Publishers, 2009. 88 p.
31. Larsson F., Felsberg M. Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition. / In Proceedings of the 17th Scandinavian Conference on Image Analysis, SCIA 2011, LNCS 6688. p. 238-249.
32. Opelt A., Fussenegger M., Pinz A., Auer P. Generic Object Recognition with Boosting. Graz University of Technology, 2004.
33. Classification: Accuracy. / Classification. URL : https://developers.google.com/machine-learning/crash-course/classification/accuracy
34. Huotari J. Why Loss and Accuracy Metrics Conflict? 2018. URL : http://www.jussihuotari.com/2018/01/17/why-loss-and-accuracy-metrics-conflict/
35. Subramanian V. Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch. Packt Publishing, Birmingham, UK, 2018. 238 p.
36. Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint: 1704.04861. 2017.
37. Iandola F. N., Han S., Moskewicz M. W., Ashraf K., Dally W. J., Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv preprint: 1602.07360. 2016.
38. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint: 1610.02357. 2017.

ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДОПОВІДІ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Дипломна робота на тему:

Згорткові нейронні мережі та їх застосування до розпізнавання дорожніх об'єктів в умовах зашумленості

Виконав:
Олексій Женчук, студент групи КА-51
Науковий керівник:
к.ф.-м.н., доц. Яковлева А.П.

Київ, 2019

Вступ

Мета роботи: дослідження можливості та властивостей застосування згорткових нейронних мереж з оптимізованим використанням параметрів до задач розпізнавання та класифікації дорожніх об'єктів в умовах наявності цифрового шуму на зображеннях, а також перевірка ефективності та порівняння алгоритмів зменшення рівня шуму в застосуванні до вказаних типів задач.

Об'єкт дослідження: набори цифрових зображень, які містять дорожні об'єкти різних класів і які можуть мати певний рівень цифрового шуму, який необхідно оцінити та відновити зображення за відповідними методами для подальшого вирішення задачі їх розпізнавання та класифікації.

Вступ

Предмет дослідження: моделі згорткових нейронних мереж з оптимізованим використанням параметрів та алгоритми зменшення шуму в зображеннях в їх застосуванні до задачі розпізнавання та класифікації дорожніх об'єктів на прикладі наборів реальних даних.

Методи дослідження: застосовані моделі згорткових нейронних мереж, методи зменшення рівня шуму в зображеннях, алгоритми навчання нейронних мереж, виконані за допомогою мови програмування Python.

Актуальність

Актуальність дослідження: розпізнавання дорожніх об'єктів є ключовим для багатьох напрямків, які швидко розвиваються, зокрема, для безпілотних автомобілів, при цьому рішення на основі задач розпізнавання та класифікації мають прийматися в реальному часі та на тільки на основі отриманих даних, в яких може бути високий рівень цифрового шуму, що значно ускладнює роботу систем. Тому необхідні моделі, здатні визначати наявність шуму та мати відповідні механізми усунення його впливу.

Постановка задачі

В роботі поставлено для виконання наступну задачу:

- провести огляд та аналіз наявних підходів до застосування згорткових нейронних мереж (ЗНМ) в задачах розпізнавання та класифікації зображень;
- розглянути механізми виникнення цифрового шуму в зображеннях та методи боротьби з ним з метою дослідити вплив алгоритмів зниження шуму на роботу згорткових нейронних мереж;

Постановка задачі

- визначити моделі ЗНМ, придатні для застосування до заданого класу задач, та проаналізувати їх роботу для чистих, зашумлених та відновлених даних та ефективність їх застосування прикладі вибраних наборів даних;
- на основі отриманих результатів проаналізувати вплив алгоритмів зниження рівня шуму на роботу згорткових нейронних мереж та зробити висновки щодо особливостей та оптимальності їх застосування.

Критерії знаходження оптимальних моделей

Критерієм оцінки ефективності роботи нейронної мережі є найменше значення функції втрат після кожної ітерації навчання моделі:

$$L(p, y) = -\sum_{n} y_n \ln p_n, n \in [1, N]$$

Але функція втрат вказує на величину розбіжності між отриманими та справжніми даними, але не кількість даних, для яких мережа робить правильні висновки щодо класу. Тому обраною метрикою оцінки результатів задачі класифікації є точність:

$$a = \frac{\sum_i c_i}{\sum_i n_i}, i \in [1, N]$$

Обраним критерієм оцінки оптимальності роботи моделей є точність на перевірочній вибірці a_{val}, яка є і критерієм для оптимального методу зменшення рівня шуму.

Моделі для згорткових нейронних мереж

Швидкість прийняття рішень для розпізнавання та класифікації дорожніх об'єктів на основі вирішення задачі класифікації має надзвичайно високе значення.

Застосування згорткових нейронних мереж з великою кількістю рівнів, які вимагають значної кількості операцій для такої сфери є неефективним.

Тому при виборі напряму розробки архітектур було вирішено зробити акцент на розгляді моделей, які пропонують ефективні моделі рівнів з меншою кількістю параметрів, при цьому зменшуючи складність моделі. Було обрано для розробки дві моделі з такого класу моделей.

Моделі для згорткових нейронних мереж

Перша модель збудована на основі ідеї мережі SqueezeNet. Головним принципом такої мережі є відмова від використання фільтрів великої розмірності, натомість використання фільтрів розмірності 1x1, які об'єднуються в окремі модулі, конфігурацією яких згодом і будується мережа.

Така мережа містить в кілька десятків разів менше параметрів, ніж моделі схожого класу.

Головним структурним елементом SqueezeNet є Fire Module, з варіантів якого будується вся система.

Моделі для згорткових нейронних мереж

Структура мережі: на рисунку зліва схема елементарного модуля, посередині та справа структура першої моделі

Моделі для згорткових нейронних мереж

Друга модель збудована на основі класу мереж MobileNet, які були запропоновані у 2017 р. для використання на пристроях з малими можливостями. Для зменшення кількості параметрів використовується ідея поглиблених розділених згорток. Ці згортки поєднують в собі послідовні операції поглибленої та поточкової згортки, які відповідно є горизонтальними та вертикальними підвидами згорток 1x1. Як наслідок, кількість параметрів на окремому рівні зменшується в кілька разів без втрати характеристик.

Модель також дозволяє задавати параметри скорочення кількості каналів та розмірності на кожному рівні мережі.

Моделі для згорткових нейронних мереж

Структура мережі: через її великий розмір вказаний елементарний модуль мережі з поглибленими та поточковими рівнями

Набір даних Traffic Signs Dataset

Набір даних Traffic Signs Dataset було створено шляхом встановлення на автомобіль цифрової камери та запису даних на 350 кілометрах автодоріг, як в населеній зоні, так і поза нею, та на автодорогах різного значення. Зображення для знаків робилися з різної відстані як кадри з відеозапису, зробленого камерою.

Загалом, отриманий набір даних для задачі класифікації містить 2193 зображень, розділених на 7 класів найпопулярніших дорожніх знаків: «Пішохідний перехід», «Об'їзд перешкоди з правого боку», «Зупинку заборонено», «Обмеження максимальної швидкості 50 км/год», «Обмеження максимальної швидкості 70 км/год», «Головна дорога» та «Дати дорогу».

Набір даних Graz02

Набір даних під назвою Graz02 було створено як набір зображень зі складними реальними характеристиками, де об'єкти класів з'являються на зображеннях в великій кількості кутів огляду та ситуацій.

Всього набір складається з 1476 зображень, які належать до чотирьох категорій основних учасників дорожнього руху: автомобілів, велосипедів, пішоходів та четвертого класу, який представляє всі інші об'єкти.

Особливостями набору даних є велика відмінність зображень для кожного представленого класу, тобто, наприклад, велика кількість різних положень та моделей автомобілів на зображеннях, а також різноманітність у фонових деталях зображення.

Набір даних Graz02

Приклад зображень з чотирьох класів для набору реальних даних

Моделювання цифрового шуму

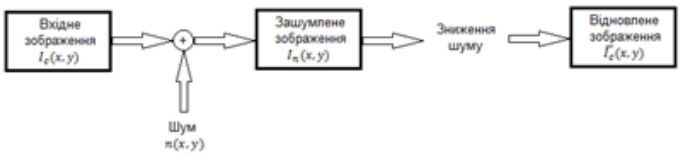
Шум є випадковими коливаннями показників елементів зображення, наприклад, значень кольорових каналів чи яскравості, які не належать до реального зображення та є його спотвореннями. Так, сенсори, що сприймають зображення, завжди мають певний рівень шуму через те, що рівень освітленості для них визначається кількістю фотонів, які потрапили на сенсор, що сама по собі є випадковим числом.

Найчастішим типом цифрового просторово незалежного шуму є гаусівський шум – такий шум, функція щільності імовірності якого відповідає нормальному (гаусівському) розподілу. Гаусівський шум зустрічається в незначній кількості в будь-якому сигналі та широко використовується для моделювання та наближення інших типів шуму. Всі практично значимі види шуму для цифрового зображення або моделюються як гаусівський, або наближуються ним та моделюються ним на практиці, або мають занадто мале поширення.

Моделювання цифрового шуму

Оцінити точність наближення, тобто різницю між зображеннями, без наявності початкового в реальних умовах неможливо, крім того, вона не характеризує придатність алгоритму до роботи з моделлю розпізнавання. Тому в якості критерія точності результатів зниження рівня шуму використовуються остаточні результати роботи моделей та відповідні метрики для різних видів шуму.

Тому для моделювання зменшення рівня шуму та його механізму використовується модель адитивного гаусівського шуму, зображена на рисунку.



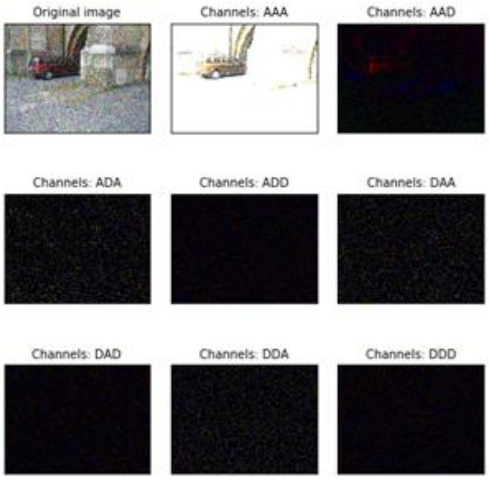
Оцінка рівня шуму

Оцінка рівня шуму є необхідною для правильної побудови та роботи алгоритмів зменшення рівня шуму. В реальних випадках, чисте (реальне) зображення невідоме, тому оцінка рівня шуму надзвичайно важлива.

Оцінити рівень для шуму, який наближено моделюється гаусівським розподілом, можна за допомогою оцінки середнього квадратичного відхилення для зображення.

Метод оцінки рівня шуму, обраний для реалізації, засновується на знаходженні деталізуючих коефіцієнтів вейвлет-перетворення.

Оцінка рівня шуму



Результати виділення шуму на зображенні з набору даних за допомогою вейвлет-перетворення

Зменшення рівня шуму

Зменшення рівня шуму має на меті відновити чисте зображення у випадку, якщо на ньому присутні зашумлені компоненти і зазвичай засновується на відомостях або припущеннях щодо того, який саме тип шуму має зображення. Це як правило зводиться до наближення екстремальних значень точок з великою різницею з оточенням до менших наближених значень в різних формах.

Як ціль знаходження методу зниження шуму було обрано такі методи, які б зберігали більшість характеристик зображення. Зазвичай, такі методи є нелінійними та враховують структурні особливості.

Зменшення рівня шуму

Серед таких методів зменшення рівня шуму було виділено:

- **метод повної варіації**, заснований на диференціальних рівняннях та полягає в мінімізації його варіації за ітеративними методами;
- метод зменшення рівня шуму за допомогою **вейвлет-перетворення** з використанням порогових значень, який базується на властивості переводу ним білого шуму в білий шум та неінформативних коефіцієнтів без шуму до нуля. Порогові значення обираються за одним із двох алгоритмів;
- метод на основі **двостороннього фільтра**, який попри порівняну простоту використовує для відновлення нелінійну комбінацію точок з врахуванням не тільки їх просторової близькості, а ще й різницю між їх характеристиками та знайшов завдяки цьому значне застосування.

Зменшення рівня шуму

Приклади результатів застосування методів зниження рівня шуму до зображень з наборів даних

Результати для набору Traffic Signs Dataset

Для набору даних Traffic Signs Dataset було реалізовано обидві з запропонованих моделей згорткових нейронних мереж. Для кожної з мереж було проведено навчання з використанням алгоритму Adam. Була застосована технологія запам'ятовування найкращої точки, коли зберігається версія моделі з найкращою метрикою (тестовою точністю). За результатами роботи було отримано такі результати:

Модель	Ориг. набір	Зашумлені дані	ВПВ	ВПV	МПВб	МПВд	ДФ
SqueezeNet	0,8269	0,5490	0,5421	0,5444	0,6720	0,7608	0,7130
MobileNet	0,7350	0,6036	0,6014	0,6082	0,6492	0,6970	0,6720

Результати для набору Graz02

Так само, як і для набору Traffic Signs Dataset, було реалізовано обидві з запропонованих моделей та проведено процес навчання за визначеною схемою з застосуванням технології запам'ятовування найкращої точки.

Для навчання моделі на основі SqueezeNet було пройдено 130 епох оновлення параметрів до появи перенавчання, тоді як для навчання моделі на основі MobileNet знадобилося 90 епох. Результати роботи наведено в таблиці:

Модель	Ориг. набір	Зашумлені дані	ВПВ	ВПV	МПВб	МПВд	ДФ
SqueezeNet	0,9728	0,9348	0,9511	0,9538	0,9592	0,9647	0,9457
MobileNet	0,875	0,8179	0,8668	0,8668	0,875	0,8697	0,8697

Висновки: моделі

Розроблені методи та моделі було застосовано до реальних наборів даних дорожніх об’єктів, де моделі показали високий рівень точності прогнозування, максимальний знайдений рівень точності для чистих даних склав 97,28 %.

Роботу згорткових нейронних мереж було перевірено для різних видів даних та різних методів зменшення рівня шуму, внаслідок чого виявлено, що одна з реалізованих моделей за нормальних умов вирішує задачу краще за другу, проте менш стійка до появи шуму на зображеннях.

Висновки: цифровий шум

Було проаналізовано вплив алгоритмів зменшення рівня шуму на результати роботи мереж та виявлено, що їх застосування в загальному випадку покращує роботу алгоритмів, інколи доводячи рівень точності близько до початкового.

Для одного з наборів даних методи з використанням вейвлет-перетворення відчутно не покращували результати, але найоптимальнішим методом зменшення рівня шуму було визначено **ітеративний метод повної варіації**.

Методи повної варіації Брегмана та метод з двостороннім фільтром також показали гарні результати.

ДОДАТОК Б ПРОГРАМНИЙ ПРОДУКТ НАВЧАННЯ

```
import tensorflow as tf
tf.test.gpu_device_name()

from tensorflow.python.client import device_lib
device_lib.list_local_devices()

from keras_applications.imagenet_utils import _obtain_input_shape
from keras import backend as K
from keras.models import Model
from keras.layers import Input, Convolution2D, MaxPooling2D, Activation, concatenate, Dropout,
warnings, GlobalAveragePooling2D, GlobalMaxPooling2D

from keras.applications.mobilenet import MobileNet
from keras.engine.topology import get_source_inputs
from keras.utils import get_file
from keras.utils import layer_utils
import h5py



sq1x1_str = "squeeze"
exp1x1_str = "expand_1x1"
exp3x3_str = "expand_3x3"

def Fire_Module(x, fire_id, squeeze=16, expand=64):
    s_id = 'fire_' + str(fire_id) + '-'

    if K.image_data_format() == 'channels_first':
        channel_axis = 1
    else:
        channel_axis = 3

    x = Convolution2D(squeeze, (1, 1), activation='relu', padding='valid', name=s_id + sq1x1_str)(x)

    left = Convolution2D(expand, (1, 1), activation='relu', padding='valid', name=s_id + exp1x1_str)(x)
```

```
    right = Convolution2D(expand, (3, 3), activation='relu', padding='same', name=s_id + exp3x3_str)(x)

    x = concatenate([left, right], axis=channel_axis, name=s_id + 'concat')
    return x


def SqueezeNet(include_top=True, weights=None,
               input_tensor=None, input_shape=None,
               pooling=None,
               classes=1000):


    input_shape = _obtain_input_shape(input_shape,
                                      default_size=227,
                                      min_size=48,
                                      data_format=K.image_data_format(),
                                      require_flatten=include_top)

    if input_tensor is None:
        img_input = Input(shape=input_shape)
    else:
        if not K.is_keras_tensor(input_tensor):
            img_input = Input(tensor=input_tensor, shape=input_shape)
        else:
            img_input = input_tensor


    x = Convolution2D(64, (3, 3), strides=(2, 2), activation='relu', padding='valid',
name='conv1')(img_input)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='pool_1')(x)

    x = Fire_Module(x, fire_id=2, squeeze=16, expand=64)
    x = Fire_Module(x, fire_id=3, squeeze=16, expand=64)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='pool_3')(x)

    x = Fire_Module(x, fire_id=4, squeeze=32, expand=128)
    x = Fire_Module(x, fire_id=5, squeeze=32, expand=128)
```

```
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='pool_5')(x)

    x = Fire_Module(x, fire_id=6, squeeze=48, expand=192)
    x = Fire_Module(x, fire_id=7, squeeze=48, expand=192)
    x = Fire_Module(x, fire_id=8, squeeze=64, expand=256)
    x = Fire_Module(x, fire_id=9, squeeze=64, expand=256)

    if include_top:

        x = Dropout(0.5, name='drop9')(x)

        x = Convolution2D(classes, (1, 1), padding='valid', activation=’relu’, name='conv10')(x)
        x = GlobalAveragePooling2D()(x)
        x = Activation('softmax', name='loss')(x)
    else:
        if pooling == 'avg':
            x = GlobalAveragePooling2D()(x)
        elif pooling=='max':
            x = GlobalMaxPooling2D()(x)
        else:
            pass

    if input_tensor is not None:
        inputs = get_source_inputs(input_tensor)
    else:
        inputs = img_input
    model = Model(inputs, x, name='squeezenet')

    return model

from keras.preprocessing import image
from keras.models import Model, load_model
from keras.layers import Dense, GlobalAveragePooling2D

from keras.initializers import RandomNormal
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K
```

```
from keras.callbacks import ModelCheckpoint
from keras.callbacks import TensorBoard
from keras.optimizers import SGD

import os.path

##SquuezeNet
##img_width, img_height = 227, 227
##MobileNet
img_width, img_height = 224, 224

img_classes = 4 ##7

base_model = SqueezeNet(include_top=False, weights=None, classes=img_classes)
x = Dropout(0.5, name='drop_9')(base_model.output)
x = Convolution2D(img_classes, (1, 1), activation='relu', padding='valid', name='conv_10')(x)
x = GlobalAveragePooling2D()(x)
x = Activation('softmax', name='loss')(x)
model = Model(input=base_model.input, output=x)
##model = SqueezeNet(weights=None, classes=img_classes)
base_model2 = MobileNet(input_shape=(224, 224, 3), include_top=False, classes=img_classes)
x = Dropout(0.5, name='drop9')(base_model2.output)
x = Convolution2D(img_classes, (1, 1), activation='relu', padding='valid', name='conv10')(x)
x = GlobalAveragePooling2D()(x)
x = Activation('softmax', name='loss')(x)
model2 = Model(input=base_model2.input, output=x)

train_data_dir = 'Graz02/train/'
validation_data_dir = 'Graz02/validation/'
nb_train_samples = 1108 ##6210
nb_validation_samples = 368 ##2076

top_epochs = 10
fit_epochs = 10

batch_size = 128 ##64, 32, 128
val_batch_size = 32 ##20
```

```
batch_size2 = 64 ##32, 128
val_batch_size2 = 32 ##20

top_layers_checkpoint_path = 'cp_top_best.h5'

## for layer in model.layers[:40]:
 ##layer.trainable = False

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) ##'rmsprop'
SGD(lr=0.01, momentum=0.9)

train_datagen = ImageDataGenerator(rescale=1. / 255)

validation_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=val_batch_size,
    class_mode='categorical')

mc_top = ModelCheckpoint(top_layers_checkpoint_path, monitor='val_acc', verbose=0,
save_best_only=True, save_weights_only=False, mode='auto', period=1)

classes_cur = validation_generator.class_indices

i = 0
for layer in model.layers:
  print(i, layer, layer.trainable)
  i+=1
```

```
model.fit_generator(
    train_generator,
    steps_per_epoch=train_generator.samples/train_generator.batch_size,
    epochs=10,##top_epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples/validation_generator.batch_size,
    callbacks=[mc_top])

from keras.utils import plot_model
plot_model(model, to_file='squeezenet_v1imgn.png', show_layer_names=True, show_shapes=True)

import shutil
shutil.copyfile(top_layers_checkpoint_path,'ckp_squimn.h5')

##model2.save('ckp_squimn_noi_7310.h5')

model_cur = load_model('ckp_squimn_8016.h5')

import numpy as np
from keras.applications.inception_v3 import preprocess_input
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

class_pred = 'bike/'

from os import listdir
from os.path import isfile, join
mypath = validation_data_dir+class_pred
onlyfiles_cur = [f for f in listdir(mypath) if isfile(join(mypath, f))]

onlyfiles_cur.sort()
print(onlyfiles_cur)
print(len(onlyfiles_cur))

i=0
j=0
for img_name in onlyfiles_cur[:10]:
```

```
  i+=1
  img_path=validation_data_dir+class_pred+img_name
  img = image.load_img(img_path, target_size=(img_width,img_height))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)
  x /= 255.
  preds = model_cur.predict(x)
  ##if bin_mod_decode_predictions(preds) == 1:
  ####img_mpl = mpimg.imread(img_path)
  ####imgplot = plt.imshow(img_mpl)
  ####plt.show()
  ####print(classes_cur)
  ##print(preds)
  for i_pred in range(len(preds[0])):
    if preds[0][i_pred] > 0.5:
      for i_cl in classes_cur:
        ####if classes_cur[i_cl] == i_pred:
        if classes_cur[i_cl] == i_pred and i_cl != class_pred[:-1]:
          j+=1
          print(i, ' Prediction for ',onlyfiles_cur[i-1])          print(classes_cur)
          print(preds)
          print(i_cl, ': ', preds[0][i_pred])
print('Not predicted ', j)
```

ДОДАТОК В ПРОГРАМНИЙ ПРОДУКТ ШУМ

```
import os
from skimage.restoration import denoise_nl_means, denoise_tv_chambolle, denoise_tv_bregman,
denoise_bilateral, denoise_wavelet, estimate_sigma
from skimage import data, img_as_float, io
from skimage.util import random_noise
import pywt
import scipy

folder_path_den = 'Graz02_noise/validation/cars/'
folder_path_new = 'Graz02_dons/validation/cars/'

def noise_level_estimate(inp_image, multichannel=False, average=False):
  if multichannel:
    num_chan = inp_image.shape[-1]
    sqvars = [noise_level_estimate(inp_image[..., ch], multichannel=False) for ch in range(num_chan)]
    if average:
      sqvar = np.mean(sqvars)
  else:
    wav_coefdet = pywt.dwtn(inp_image, wavelet='db2')['d'*inp_image.ndim]
    nz_wav_coefdet = wav_coefdet[np.nonzero(wav_coefdet)]
    quan = scipy.stats.norm.ppf(0.75)
    sqvar = np.median(np.abs(nz_wav_coefdet)) / quan
  return sqvar

onlyfiles = [f for f in os.listdir(folder_path_den) if os.path.isfile(os.path.join(folder_path_den, f))]
onlyfiles.sort()
print(onlyfiles)
print(len(onlyfiles))

i_im = 0
for image_name in onlyfiles:
    print(i_im, ' Image ', image_name)
    original = img_as_float(io.imread(folder_path_den+image_name))
    ####img_mpl = mpimg.imread(folder_path_den+image_name)
    ####imgplot = plt.imshow(img_mpl)
    ####plt.show()
```

```
    sigma_est = noise_level_estimate (original, multichannel=True, average =True)

    print(sigma_est)
    if sigma_est > 0.1:
        tv_weight = 0.09
    elif sigma_est > 0.07:
        tv_weight = 0.07
    elif sigma_est > 0.04:
        tv_weight = 0.03
    elif sigma_est > 0.01:
        tv_weight = 0.02
    if sigma_est > 0.01:
        if not os.path.exists(folder_path_new):
            os.makedirs(folder_path_new)
        newpath = folder_path_new + image_name
        ##nl_means
        io.imsave(newpath, denoise_nl_means(original))
        ##tv_bregman
        ##io.imsave(newpath, denoise_tv_bregman(original, weight=1/tv_weight))
        ##tv_chambolle
        ##io.imsave(newpath, denoise_tv_chambolle(original, weight=tv_weight, multichannel=True))
        ##bilateral
        ##io.imsave(newpath, denoise_bilateral(original, multichannel=True))
        ##wavelet
        ##io.imsave(newpath, denoise_wavelet(original))
        ####img_mpl = mpimg.imread(newpath)
        ####imgplot = plt.imshow(img_mpl)
        ####plt.show()

    else:
        if not os.path.exists(folder_path_new):
            os.makedirs(folder_path_new)
        newpath = folder_path_new + image_name
    i_im += 1

##Recognition in real-time
```

```
from os import listdir
from os.path import isfile, join

import numpy as np
from keras.preprocessing import image
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K

from keras.applications.inception_v3 import preprocess_input
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
model_cur = load_model('ckp_squimn_8016.h5')



mypath = validation_data_dir+class_pred ##mypath = test_data_dir+class_pred
onlyfiles_cur = [f for f in listdir(mypath) if isfile(join(mypath, f))]
onlyfiles_cur.sort()
print(onlyfiles_cur)
print(len(onlyfiles_cur))

def denoise_method(original_image, dnmethod='tv_iter_chambolle', weight=0.1, multichannel=False,
method=''):
  if dnmethod == 'wavelet_bv':
    return denoise_wavelet(original, method=method)
  if dnmethod == 'tv_bregman':
    return denoise_tv_bregman(original_image, weight=1/weight)
  if dnmethod == 'tv_iter_chambolle':
    return denoise_tv_chambolle(original_image, weight=weight, multichannel=multichannel)
  if dnmethod == 'dn_bilateral':
    return denoise_bilateral(original, multichannel=multichannel)

i=0
j=0
show_im = False
for img_name in onlyfiles_cur:
  i+=1
```

```
  img_path=test_data_dir+class_pred+img_name ##img_path=validation_data_dir+class_pred+img_name
  original = img_as_float(io.imread(img_path))

  if show_im:
    img_mpl = mpimg.imread(img_path)
    imgplot = plt.imshow(img_mpl)
    plt.show()
  sigma_est = noise_level_estimate (original, multichannel=True, average =True)
  print(sigma_est)
  if sigma_est > 0.1:
    tv_weight = 0.09
  elif sigma_est > 0.07:
    tv_weight = 0.07
  elif sigma_est > 0.04:
    tv_weight = 0.03
  elif sigma_est > 0.01:
    tv_weight = 0.02
  if sigma_est > 0.01:
    if not os.path.exists('TSD/testing/'):
      os.makedirs('TSD/testing/')
    newpath = 'TSD/testing/' + img_name
    io.imsave(newpath, denoise_method(original, dnmethod='tv_iter_chambolle', weight=tv_weight,
multichannel=True))
    ####img_mpl = mpimg.imread(newpath)
    ####imgplot = plt.imshow(img_mpl)
    ####plt.show()
  img = image.load_img(newpath, target_size=(img_width,img_height))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)
  x /= 255.

  preds = model_cur.predict(x)
  if show_im:
    img_mpl = mpimg.imread(newpath)
    imgplot = plt.imshow(img_mpl)
    plt.show()
    print(classes_cur)
```

```
    print(preds)
  os.remove(newpath)
  for i_pred in range(len(preds[0])):
    if preds[0][i_pred] > 0.5:
      for i_cl in classes_cur:
        if classes_cur[i_cl] == i_pred and i_cl != class_pred[:-1]:
          j+=1
          print(i, ' Prediction for ',onlyfiles_cur[i-1])
          ##img_mpl = mpimg.imread(img_path)
          ##imgplot = plt.imshow(img_mpl)
          ##plt.show()
          print(classes_cur)
          print(preds)
          print(i_cl, ': ', preds[0][i_pred])
print('Not predicted ', j)
```