DOI: https://doi.org/10.15407/usim.2018.06.046
УДК 581.513

Р.М. ТИМЧИШИН, аспірант, відділ інтелектуального управління,
Міжнародний науково-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Глушкова, 40, Київ 03187, Україна,
romantymchyshyn.rt@gmail.com

О.Є. ВОЛКОВ, завідувач відділу інтелектуального управління,
Міжнародний науково-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Глушкова, 40, Київ 03187, Україна,
alexvolk@ukr.net

О.Ю. ГОСПОДАРЧУК, старший науковий співробітник, відділ інтелектуального управління,
Міжнародний науково-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Глушкова, 40, Київ 03187, Україна,
dep185@irtc.org.ua

Ю.П. БОГАЧУК, провідний науковий співробітник, відділ інтелектуального управління,
Міжнародний науково-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Глушкова, 40, Київ 03187, Україна,
dep185@irtc.org.ua

СУЧАСНІ ПІДХОДИ ДО РОЗВ'ЯЗАННЯ ЗАДАЧ КОМП'ЮТЕРНОГО ЗОРУ

Наведено структурований огляд сучасних методів розв'язання задач комп'ютерного зору, їх переваг та недоліків, і визначення невирішених проблем. Цей напрямок швидко прогресує, що пов'язано зі збільшенням обчислювальної потужності комп'ютерів, а також підключенням до дослідження таких гігантів IT індустрії, як Google і Microsoft. Розглянуто абсолютно різні за своєю природою підходи: підхід на основі нечіткої логіки; підхід на базі згорткових нейронних мереж та глибокого навчання; підхід з використанням детекторів і дескрипторів. Розглядається не тільки точність алгоритмів, але і їх швидкодія та затрати пам'яті, що відіграють важливу роль для вбудованих систем (безпілотних літальних апаратів, мобільних пристроїв, роботизованих та супутникових систем).

Ключові слова: *комп'ютерний зір, класифікація зображень, виявлення об'єктів, сегментація зображень, фільтрація зображень, ідентифікація контурів, нечітка логіка, нейронні мережі, детектори, дескриптори.*

Вступ

Комп'ютерний зір — надзвичайно широка область, яка включає в себе багато різнопланових задач, таких яких сегментація, фільтрація, класифікація, реконструкція сцени, оцінка положення об'єкта, виявлення об'єктів, відеоспостереження та багато інших. Комп'ютерне бачення є важливою складовою розвитку штучного інтелекту та інтелектуальних інформаційних технологій.

Комп'ютерне бачення використовується в десятках галузей, наприклад, при побудові «розумних» магазинів, ідентифікації клієнтів за допомогою біологічних характеристик, автоматизації сільськогосподарських процесів з використанням дронів, автоматичній інспекції на виробництвах, відеоспостереженні, покращенні якості фото- та відеоданих (фільтрація), автоматичній доставці посилок безпілотними літальними апаратами. Коло застосувань цієї технології розширюється, адже потреба у сис-

темах штучного інтелекту зростає, а зір — це один з найінформативніших сенсорів, який може використовуватись в таких системах.

І хоча значного прогресу в цій галузі вже досягнуто, залишається багато нерозв'язаних задач. Існуючим алгоритмам бракує загальності, а збільшення швидкодії зазвичай викликає зменшення точності. Тому актуальними напрямками є: покращення швидкодії існуючих алгоритмів; застосування алгоритмів розпізнавання на вбудованих системах, які мають обмежені ресурси пам'яті і обчислювальні потужності, наприклад, безпілотні літальні апарати, роботизовані та супутникові системи; розв'язання задач в режимі реального часу в умовах обмежених ресурсів; покращення точності; зменшення затрат на навчання систем, що базуються на нейронних мережах; розширення кола об'єктів розпізнавання; розпізнавання зображень низького розширення та якості.

В роботі аналізуються основні підходи до розв'язання задач комп'ютерного зору, висвітлено їх переваги та недоліки.

Комп'ютерний зір з використанням нечіткої логіки

Задачі комп'ютерного зору передбачають певний рівень невизначеності. Це є основною причиною використання теорії нечіткої логіки.

Вважається, що значного покращення в задачах комп'ютерного бачення можна досягнути, якщо збільшити кількість інформації, яку можна обробити за адекватний період часу. Але хоча людина не вміє обробляти величезні потоки даних в реальному часі, вона виконує задачу розпізнавання дуже і дуже успішно. Вміння підбирати потрібний рівень деталізації — ось що дозволяє людині розпізнавати об'єкти, занадто загальний опис може призвести до пропуску важливих деталей та відвести увагу від основних характеристик. Теорія нечіткої логіки дозволяє варіювати цим рівнем узагальнення змінюючи кількість лінгвістичних змінних в системі та варіюючи вигляд функцій належності нечітких множин.

Системи комп'ютерного зору повинні вміти представляти невизначеність та передбачати ефекти невизначеності для правильної інтерпретації отриманих результатів.

Невизначеність часто розглядається як результат деякого випадкового процесу, проте в комп'ютерному баченні невизначеність може виникати і з інших причин, серед яких: проекція зображень у простір меншої розмірності, зміна освітлення, дискретизація просторових чи часових (у випадку відео) координат, невідома якість зображення, неточні обчислення, перетин класів розпізнаваних об'єктів, тобто неможливість чітко сформулювати ознаки і визначення об'єктів. Теорія нечіткої логіки ідеально підходить для розв'язання проблем невизначеності такого роду.

Нечітка логіка дозволяє легко переносити накопичений досвід в системи комп'ютерного бачення, використовуючи прості та зрозумілі правила. Людина, здійснюючи класифікацію зображень, не працює з суто числовими характеристиками об'єктів, які зображені, натомість ми класифікуємо об'єкт за базою правил, які отримані з досвідом і можемо легко сформулювати. Це є великою перевагою таких систем, оскільки потенціал для їх покращення практично невичерпний і завжди можна сформулювати правила більш жорстко, чи доповнити систему новими, уточнюючими, правилами.

Базовим поняттям нечіткої логіки є нечітка множина. Нечітка множина — це множина, елементи якої належать їй в певній мірі, на відміну від традиційних множин, де елементи або належать множині або ні. Нечітка множина — це пара: (U, m), де U — множина, а $m: U \to [0,1]$ — функція належності або характеристична функція множини. Нечіткі множини допускають більшість операцій, які можна виконувати зі звичайними множинами, наприклад доповнення, перетин і об'єднання.

Вважаючи, що елемент належить нечіткій множині в певній мірі, судження в нечіткій логіці є правдивими теж в певній мірі, на відміну від традиційної логіки, де висловлювання може бути або вірним, або хибним.

Нечітким k-арним предикатом є функція $P(x_1, x_2, ..., x_k)$: $X_1 \times X_2 \times ... \times X_k \to [0,1]$ — відображення декартового добутку універсумів на

відрізок [0,1]. З нечіткими висловлюваннями можна виконувати багато операцій традиційної логіки: заперечення, кон'юнкція, диз'юнкція, імплікація, еквівалентність тощо.

Під правилом нечіткої продукції або під нечіткою продукцією зазвичай розуміють вираз наступного вигляду:

$$(i): Q;\ P;\ \tilde{A} => \tilde{B};\ S,\ F,\ N,$$

де (i) — ім'я нечіткого правила, певна сукупність символів, яка дозволяє однозначно ідентифікувати правило;

Q — сфера застосування нечіткого правила;

P — умова застосовності ядра нечіткого правила;

N — постумова нечіткого правила;

$\tilde{A} => \tilde{B}$ — ядро правила, центральний компонент продукції, зазвичай записується у вигляді «якщо ..., то ...»;

S — метод визначення кількісного значення міри істинності висновку $\tilde{B}$ на основі відомого значення істинності $\tilde{A}$ (прямий і обернений, або *FMP — fuzzy modus ponens* та *FMT — fuzzy modus tollens*);

F — коефіцієнт впевненості, який визначає кількісну оцінку міри істинності, часто називається ваговим коефіцієнтом правила.

Існує декілька типів нечітких множин. Перший тип пов'язує з множиною функцію, яка має досить конкретні значення, тобто є чіткою, і багато дослідників критикують теорію нечітких множин через це. Тому запропоновано новий тип нечітких множин. Нечіткі множини другого типу зазвичай поділяють на інтервальні (IT2 — *interval type*-2) та загальні (*GT*2 — *general type*-2). Загалом особливістю множин другого типу є те, що функція належності також набуває певної нечіткості. Вводиться поняття сліду невизначеності — об'єднання всіх функцій належності. У випадку інтервальної нечіткої множини вводиться дві функції — верхня і нижня функції належності і слід невизначеності виглядає наступним чином:

$$FOU(\textit{footprint of uncertainty})(A)=\cup_{\forall x\in X}\left[\underline{\mu}(x),\overline{\mu}(x)\right].$$

Нечіткі множини загального типу вносять ще один рівень свободи і їх функція належності є уже 3-вимірною. Слід невизначеності таких множин виражають так:

$$FOU(\tilde{A})=\{(x,u)\in X\times[0,1]\mid \tilde{\mu}(x,u)>0\}.$$

Для обмеження складності логіки загального типу часто використовують концепцію α-площин, слід невизначеності яких виглядає наступним чином:

$$FOU(\tilde{A}_\alpha)=\{(x,u)\in X\times[0,1]\mid \tilde{\mu}(x,u)\geq\alpha\}.$$

Основними кроками проектування систем на базі нечіткої логіки є:

1) фаззіфікація — підбір лінгвістичних змінних, визначення потрібного рівня деталізації;

2) підбір функцій належності для кожної нечіткої множини кожної лінгвістичної змінної;

3) вибір бази нечітких правил таким чином, щоб мінімізувати їх кількість та максимізувати точність;

4) дефаззіфікація — приведення виходу системи до чіткого вигляду, лінгвістичні змінні, які використовуються як вихід системи часто неможливо прямо застосувати для розв'язання задачі, зазвичай їх потрібно перевести в кількісний вигляд (цей крок часто називають редукцією, найпоширенішим методом є редукція з використанням центру множини).

Нечіткі множини можуть використовуватись для опису понять типу «близько»/«далеко», а також для визначення чітких областей зображення, що відповідають поняттю «північно-східний».

Просторове розташування об'єктів один відносно іншого має велике значення в задачах комп'ютерного бачення і, хоча людина легко може описати його, дуже важко навчити автоматичну систему розв'язувати цю задачу саме через нечіткість визначень. Просторове відношення об'єктів і його зв'язок з розумінням людиною зображеного, є дуже важливим, тому дослідники розглядають цю проблему з точки зору лінгвістики та психології. Просторові відношення «над» чи «зліва» не мають точних визначень і тому є хорошим прикладом для застосування нечітких множин. Проте суб'єктивність і складність цих концепцій зумовлює складність

об'єктивного визначення просторових відношень, що зумовило наявність великої кількості нечітких визначень.

Комп'ютерне бачення — досить широка область, яка включає в себе багато різних задач. Розглянемо деякі з цих задач в контексті нечіткої логіки.

Сегментація зображень. Як відомо, сегментація зображень — процес виділення значущих сегментів на зображенні з метою покращення подальшого виявлення об'єктів, опису сцени і розуміння контексту. Процес знаходить області зі спільними ознаками, такими як текстура, колір та ін. Саме визначення не є чітким: невідомо як визначити схожість між пікселями чи їх областями. Сегментація зображень є одним із найважливіших завдань комп'ютерного бачення, це один з перших кроків класифікації. Якщо зображення буде неправильно поділене на області, то і їх подальша класифікація буде невірною. Результати сегментації можуть розглядатися як нечіткі множини. Кожній області присвоюється нечітка множина і визначається рівень з яким кожен піксель належить кожній множині. Після цього для отримання остаточного результату за традиційними техніками порогових значень, кластеризації, сегментації з учителем та сегментації на базі правил застосовуються техніки теорії нечітких множин.

В [1] розглянуто проблему сегментації дистанційних зображень таких як знімки з висоти пташиного польоту. Вони використовують нечіткі множини першого та другого типів. Гаусівські моделі першого типу використовуються для моделювання невизначеності, що присутня на зображенні. В подальшому нечітка модель другого інтервального типу будується шляхом «розмивання» математичного сподівання та дисперсії, в результаті чого отримується верхня та нижня функції належності. Запропонована модель другого типу допомагає підсилити вираження невизначеності на зображенні та одночасно зменшити невизначеність у системі прийняття рішення. Далі нечітка функція належності з її верхньою та нижньою функціями належності використовується як вхід нейронної мережі, що є системою прийняття рішень. По суті нечіткі моделі використовуються для покращення роботи фінальної моделі — нейромережі.

В [2] автори застосовують нечіткі множини другого інтервального типу в моделі активного контуру для покращення сегментації. Вони здійснюють ітераційне оновлення пікселів тільки у вузькій смузі вздовж контуру через значне збільшення обчислювальних затрат при використанні нечіткої логіки другого типу.

Загалом, методи нечіткої логіки використовуються при розв'язанні задач сегментації як допоміжний засіб для покращення існуючих алгоритмів. Потрібно зазначити, що застосування нечітких множин другого типу може значно збільшити обчислювальні затрати. Існуючі роботи спрямовані на розв'язання задачі сегментації з застосуванням нечіткої логіки зазвичай не використовують якісь спільні бази даних, що робить порівняння їх з іншими алгоритмами практично неможливим. Крім того всі вони використовують різні метрики для оцінки результатів роботи алгоритму, як то візуальна оцінка результатів, кількість згенерованих сегментів, відсоток помилок тощо.

Фільтрація зображень. Часто отримані зображення є спотвореними або містять шум, який ускладнює процес розпізнавання. Фільтрація — це набір інструментів, який глушить або підкреслює певні види інформації на зображенні. Через нечітку природу цієї задачі, ряд робіт присвячено саме підходам на базі нечіткої логіки.

В [3] використано алгоритм на базі IT2 *FLS* (*interval type*-2 *fuzzy logic system*). Запропонований фільтр складається з двох підфільтрів. Один фільтр обчислює відстань між компонентами кольору центрального пікселя та його околу. Ця відстань визначає в якій мірі компонента повинна бути відкоригована. Для кожного пікселя визначаються пари $rg(i, j)$ — *red/green*, $rb(i, j)$ — *red/blue*, $gb(i, j)$ — *green/blue*. Щоб відфільтрувати піксель в позиції (i, j) використовується рамка 3×3 і для кожної пари обчислюються відстані між центральним пікселем та іншими пікселями з рамки. Далі кожному пікселю, на основі нечітких правил, присвоюється певна вага. Щоб

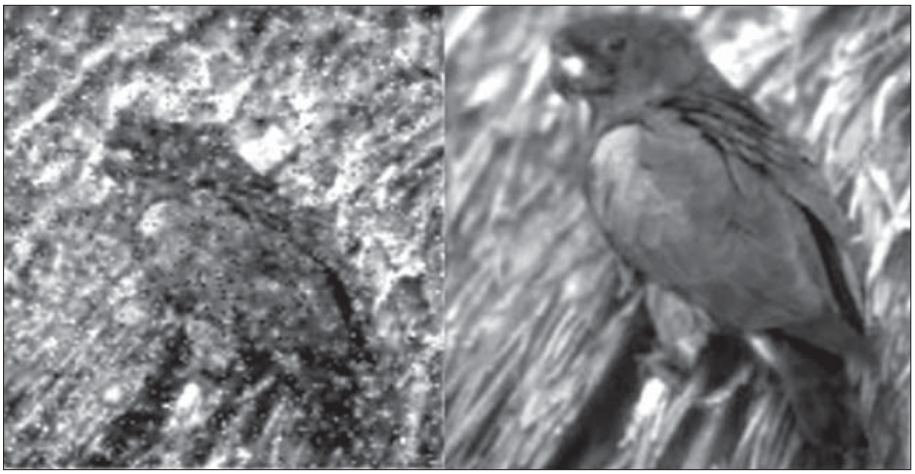


Рис. 1. Результат роботи алгоритму фільтрації

визначити чи відстань між компонентами велика чи мала, використовують нечіткі множини "*small*" для кожної пари, що характеризуються функціями μ_{rgs}, μ_{rbs}, μ_{gbs}, яким відповідають верхня і нижня функції належності, оскільки розглядається IT2 (*interval type*-2) логіка. Інший фільтр використовує попередній результат, щоб обчислити локальну відстань для кожної компоненти кольору окремо, і для кожного пікселя продукує кількісне значення, яке потрібно відняти від кожної компоненти кольору цього пікселя, щоб прибрати шум.

Приклад роботи алгоритмів можна побачити на рис. 1.

В [4] введено імпульс—детектор, який використовує IT2 *FLS*. Детектор базується на селективному фільтруванні використовуючи спеціальний оператор, який вирішує, чи піксель потребує фільтрування. Внутрішні параметри функцій належності виводяться під час тренування. Алгоритм тестувався з трьома типами шумів.

Запропонована в [5] адаптивна T2 нечітка медіанна фільтрація дозволяє пом'якшувати імпульсний шум зберігаючи при цьому деталі зображення. Фільтр використовує здатності нечітких систем управління невизначеністю другого типу в комбінації з медіанним фільтром і фільтром Аракави. Алгоритм зменшує затрати пам'яті у порівнянні з іншими алгоритмами фільтрації, що базуються на нечіткій логіці другого типу.

Ідентифікація контурів та кутів (edge detection). Однією з найбільш важливих характеристик зображення є контури і ребра, які ідентифікують розташування об'єктів і описують їх форму. В ідеальному зображенні ребра відповідають контурам об'єкта і тому є ефективним підходом до сегментації зображень. Проте визначення ребра є досить розмитим. Інтуїтивно зрозуміло, що точки, в околі яких є значний скачок інтенсивності, скоріш за все належать ребру. Проте невідомо як визначити рівень за якого стрибок в інтенсивності визначає ребро, а за якого ні. Або як визначити, що саме цей піксель належить ребру, а не його сусід. Зрозуміло, що таке нечітке визначення краще всього представляти за допомогою нечітких множин.

В [6] запропоновано метод, що базується на техніці морфологічного градієнта та IT2 *FLS*. Система нечіткої логіки спроектована з використанням гаусівських функцій належності, а параметри обчислюються динамічно на базі значень градієнта кожного зображення. Система складається з 4 входів, одного виходу та трьох нечітких правил, сформованих на базі декількох тестів та експертних знань. Редукція проводилась використовуючи центр множин.

В [7], так само як [6], використовується техніка морфологічного градієнта, проте в комбінації з *GT*2 *FS* (*general type*-2 *fuzzy set*). Система має чотири входи з трьома лінгвістичними

змінними і один вихід з двома лінгвістичними змінними. Використовується структура правил Мамдані, дефаззіфікація проводиться використовуючи методи висот та наближень. Для тестування використовувалась база даних [6] і одне синтетичне зображення. До зображень застосовано гаусівський шум. Множину порівняльних досліджень представлено в [6]: слід невизначеності *GT*2 функцій належності варіювався з метою покращення результатів; дефаззіфікація проводилась за допомогою методу висот та методу наближень, де метод висот давав кращі результати на зображеннях без шуму, а метод наближень — на зображеннях з гаусівським шумом; кількість α-площин варіювалась з метою знаходження площин потрібних для апроксимації результату; проведено дослідження з використанням традиційного методу морфологічного градієнта, методу на базі *Type*-1 *FL* (*type*-1 *fuzzy logic*), а також IT2 *FL* та *GT*2 *FL*. Методи на базі *GT*2 *FL* мають найкращі результати, що автори пояснюють більшою кількістю степенів свободи.

В [8] автори, на відміну від інших робіт, застосовують ідентифікацію контурів на зображеннях у кольорі. Запропонований підхід поєднує методи, що базуються на градієнтах і *GT*2 *FLS*. Системи нечіткої логіки є наближеними та використовують α-площини. Для тестування використовувались зображення у кольорі з шумом та без. Використання кольорових зображень очевидно потребує більших обчислювальних затрат, проте кількість інформації в кольоровому зображенні набагато більша, ніж в чорно-білому, що потенційно може покращити ідентифікацію об'єктів та розуміння сцени і контексту.

Результат роботи градієнтного методу ідентифікації контурів на основі градієнта на кольоровому зображенні наведено на рис. 2.

Загалом, дослідники, працюючи з абсолютно різними базами даних, не використовують спільної еталонної бази, а оцінюють результати роботи за різними критеріями, як наприклад: візуальна оцінка, критерій Пратта, відсоток правильних класифікацій тощо, тому порівняти алгоритми між собою — досить складно. Варто зазначити, що алгоритми на основі *GT*2 *FS* (*general type*-2 *fuzzy set*) вважаються найбільш точними, проте потрібно брати до уваги обчислювальну складність алгоритмів, які базуються на цій теорії. Особливо гостро ця проблема постає у випадках, коли задача розв'язується в режимі реального часу.

Класифікація зображень — процес присвоєння об'єктам на зображенні міток певних класів.

В [9] запропоновано класифікатор на базі *GT*2 *FS* для застосування в класифікації земельного покриву (рівнинної культивованої території) за зображеннями з висоти. Для класифікації використовували картограф із 7 діапазонами довжин хвиль. Задача полягає в розпізнаванні областей з висадженою соєю, ку-

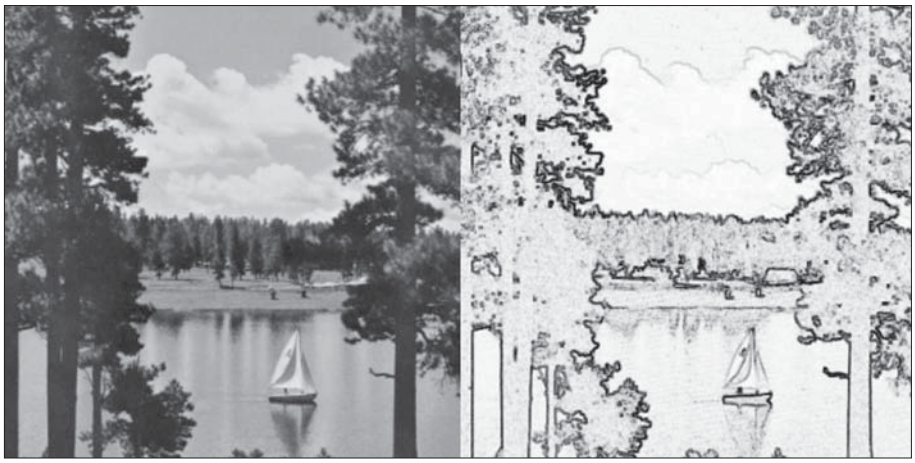


Рис. 2. Результат роботи градієнтного методу ідентифікації контурів

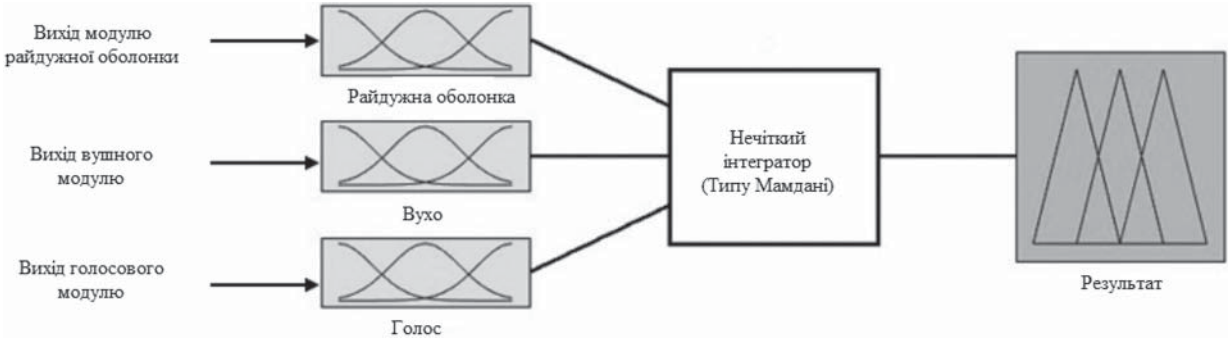


Рис. 3. Схема системи розпізнавання людей на базі нечіткої логіки

курудзою та люцерном. Автори використали відповідні лінгвістичні змінні: «соя, кукурудза чи люцерн» для кожного діапазону довжини хвилі. Запропонований метод на базі *GT*2 *FS* перевершив методи на базі інших типів логік, проте все ще поступається методу максимальної правдоподібності. Автори стверджують, що такий метод краще застосовувати для більш складних баз даних, оскільки головною перевагою методу є вміння справлятись з невизначеністю.

Багато робіт пов'язаних з класифікацією присвячено розпізнаванню медичних знімків, як наприклад, рентгенографічні знімки гомілки чи легенів або мікроскопічні знімки клітин.

Загалом, кількість робіт за темою класифікації, що використовують теорію нечіткої логіки, значно менша, ніж кількість робіт за темою сегментації, фільтрації чи виявлення ребер.

Підбір параметрів нечіткої системи з використанням генетичних алгоритмів. Для розв'язання задач розпізнавання образів застосовують генетичні алгоритми. Цікавим є поєднання генетичних алгоритмів та нечіткої логіки для розв'язання цієї задачі, наприклад, у [10] розглянуто проблему ідентифікації людини на основі її райдужної оболонки, голосу та вуха. Тут нечітка логіка використовується для комбінування результатів модульної нейронної мережі. Модульна нейронна мережа — це такий тип нейронної мережі, де обчислення розділені на декілька модулів, кожен з яких збирає інформацію певного типу (в даному випадку — райдужна оболонка, вухо та голос). На рис. 3 можна побачити схематичне зображення системи, представленої в [10].

Результати, отримані модульною нейронною мережею подаються на вхід системи на основі нечіткої логіки, де вони комбінуються на основі бази правил для отримання остаточного результату.

Представлена система використовує генетичний алгоритм для оптимізації практично всіх параметрів системи: тип нечіткої логіки (перший, другий інтервальний та загальний), тип системи (*Mamdani* чи *Sugeno*), тип функцій належності (трапезоїд чи *GBell*), кількість функцій належності в кожній вхідній та вихідній змінній, їх параметри, а також правила.

Використовуючи таку оптимізацію вдалося добитись розпізнавання на рівні 99—100% у випадках без шуму та в середньому 90% розпізнавання у випадках з шумом (найгірші випадки — 83%, найкращі — 99%).

Таке поєднання генетичних алгоритмів та нечіткої логіки є перспективним і потенційно може застосовуватись для класифікації об'єктів.

Класифікація та виявлення об'єктів за допомогою нейронних мереж

Основними інструментами, на яких базуються сучасні алгоритми класифікації зображень є *Deep Learning* та *CNN* (*convolutional neural networks* — згорткові нейронні мережі).

Згорткові нейронні мережі схожі з звичайними нейронними мережами, проте вони спроектовані з припущенням, що на вхід подається

зображення. Це дозволило закласти певні характеристики в архітектуру, які спрямовані на покращення класифікації. На відміну від звичайних нейронних мереж, шари згорткових нейронних мереж є тривимірними. Нейрони одного шару пов'язані тільки з певною областю наступного шару. Перший вхідний шар мережі має розмірність $w \times h \times d$, де w — ширина зображення, h — висота, d — кількість каналів кольору [11]. Кожен шар такої мережі трансформує отриманий вхід, використовуючи диференційовну функцію. В згортковій нейронній мережі використовуються такі типи шарів: *CONV* (*convolutional* — згорткові), *POOL* (*pooling* — агрегуючі), *FC* (*fully-connected* — повнозв'язні).

Параметри згорткового шару — множина фільтрів. Типовий фільтр може мати розмірність $5 \times 5 \times 3$. Під час прямого проходу цей фільтр рухається по вхідному шару і обчислює добуток між елементами фільтру та областю вхідного зображення. Таким чином формується так звана активаційна карта, що містить відповідь фільтра на вхідний шар в кожній позиції. Інтуїтивно зрозуміло, що в процесі навчання формуються фільтри, які реагують на певні характеристики, наприклад, ребро, пляма певного кольору тощо. Насправді проблема інтерпретації роботи нейронної мережі є складною і не вирішеною на даний момент, але така аналогія дозволяє краще уявити принцип роботи. Для кожної області вхідного шару використовується декілька фільтрів, кожен з яких потенційно повинен розпізнавати різні характеристики цієї області. Кожен фільтр продукує двовимірну активаційну карту, які потім нашаровуються одна на одну вздовж глибини вихідного шару. Розглянуту концепцію можна наглядно показати на рис. 4. Вхідний шар (червоний) має розмірність $32 \times 32 \times 3$, тобто на вхід подається зображення 32×32 з трьома каналами кольору. Кожен нейрон згорткового шару зв'язаний тільки з певною областю вхідного шару, проте з повною глибиною. Наведено приклад згорткового шару з використанням 5 фільтрів для кожної області.

Варто зазначити, що фільтри можуть рухатись з різними кроками (1, 2 і т. ін.) тим самим ми можемо варіювати розмірність вихідного шару, чим більший крок, тим менша буде розмірність шару. Інколи до вхідного зображення дописують нульові елементи по краях, щоб стандартизувати всі зображення до певного розміру на який розрахована нейронна мережа.

Така схема все ще містить дуже багато нейронів і потребує багато обчислень. Тому дійшли висновку, що якщо є сенс шукати певну характеристику в одному місці зображення, то є сенс шукати її і в інших місцях. Тому вирішено для нейронів на певній глибині використовувати один і той самий фільтр для кожної просторової позиції, що значно зменшує кількість унікальних параметрів шару.

Між згортковими шарами таких мереж час від часу вставляють агрегуючі шари для того, щоб контролювати розмірність вихідного шару. Такі шари трансформують тільки ширину та висоту вихідного шару. Вони складаються з фільтрів розміру $k \times l$, які агрегують значення в області замінюючи всі значення цієї області одним. Прикладом такої агрегуючої операції є MAX. Такий фільтр залишає тільки максимальне значення, наприклад, в області 2×2 він відкидає 75% параметрів, залишаючи одне з чотирьох значень. Застосування таких шарів дозволяє зберегти важливу інформацію тим самим зменшивши кількість параметрів в системі. Приклад такого шару можна побачити на рис. 5.

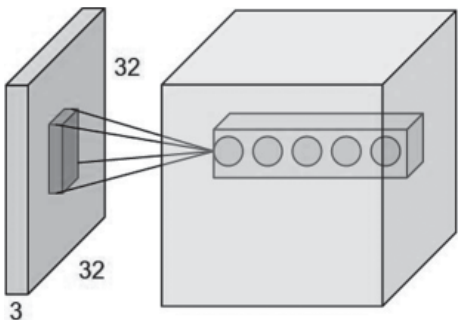


Рис. 4. Приклад згорткового шару нейронної мережі

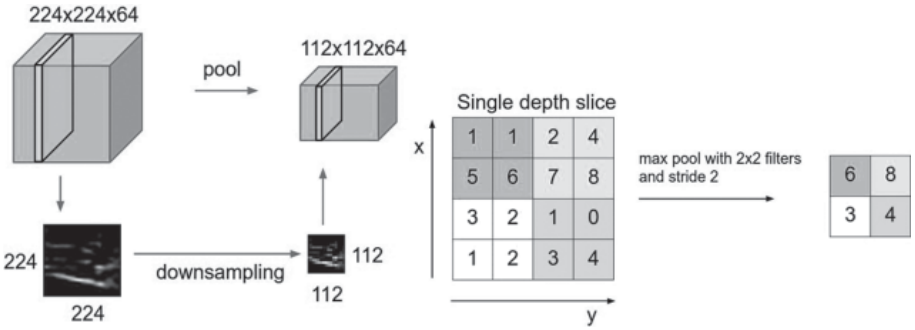


Рис. 5. Агрегуючий шар згорткової нейронної мережі

Нейрони повнозв'язного шару з'єднані з усіма нейронами вхідного шару як і у звичайних нейронних мережах.

Є багато відкритих баз зображень та відео на яких розробники алгоритмів порівнюють свої напрацювання. Такими базами є *CIFAR*-10, *CIFAR*-100, *ILCVRC* (*ImageNet Large Scale Visual Recognition Challenge*), *PASCAL VOC Challenge, COCO* (*Common Objects in Context*).

Класифікація зображень. AlexNet [12] — перша робота, що популяризувала використання згорткових нейронних мереж в задачах комп'ютерного бачення. Виграла *ILSVRC* в 2012 з результатом 18.9% top 5 error. Особливістю мережі є те, що згорткові шари розташовані один за одним. До цього, зазвичай, використовували згорткові шари, за якими одразу йшли агрегуючі шари. Мережа складалась з 60 млн. параметрів, 500 000 нейронів, п'ять згорткових шарів, деякі з яких передували шарам з max-агрегацією, два повнозв'язних шари та фінальний 1000-вимірний softmax шар. Ця архітектура ефективно реалізована на *GPU* з застосуванням регуляризації для уникнення перенавчання. Все це дуже пришвидшило як навчання так і класифікацію.

ZFNet [13] — згорткова нейронна мережа, що є покращенням *AlexNet* (14.8% *top* 5 *error* на *ILSVRC* 2013). Вона використовує схожу архітектуру, але інші гіперпараметри, зокрема збільшений розмір внутрішніх згорткових шарів, а також зменшений крок і розмір фільтра на першому шарі.

VGGNet [14] — мережа, основним здобутком якої є доведення емпіричним шляхом того, що глибина мережі є критичним моментом для хороших показників. Значного покращення можна досягнути, використовуючи 16—19 згорткових / повнозв'язних шарів з дуже однорідною архітектурою, яка виконує лише 3×3 згортки та 2×2 агрегацію. Недоліками є те, що вона повільна та використовує багато пам'яті та параметрів (140 млн.). Більшість параметрів знаходяться на першому повнозв'язному шарі. Пізніше доведено, що ці повнозв'язні шари можуть бути видалені без зниження якості розпізнавання, при цьому суттєво зменшуючи кількість необхідних параметрів. *VGGNet* ще й зараз використовується в якості мережі для формування карти ознак (feature extraction).

GoogLeNet (*Inception-v*1) [15] — вперше представила архітектуру *Inception* (6.67% *top* 5 *error* на *ILSVRC* 2014), що значно зменшила кількість параметрів в мережі (4 млн.). Мережа складається з дев'яти блоків або *Inception* модулів, схожих за архітектурою. Кожен з блоків робить згортки 1×1, 3×3, 5×5 та *max*-агрегацію. Таким чином на кожному рівні витягуються ознаки різного масштабу. Якщо масштаб занадто великий для поточного рівня — він розпізнається на наступному рівні.

Після *Inception* модулів йде агрегація за середнім та softmax шар. Для пришвидшення тренування використовуються ще допоміжні так звані training heads, тобто на проміжних рівнях додається вихід, який дозволяє сигналу швидше доходити до нижніх рівнів та є додатковою регуляризацією.

ResNet [16] — основна ідея таких мереж полягає у введенні в архітектуру так званих обхідних з'єднань, які дозволяють просто пропус-

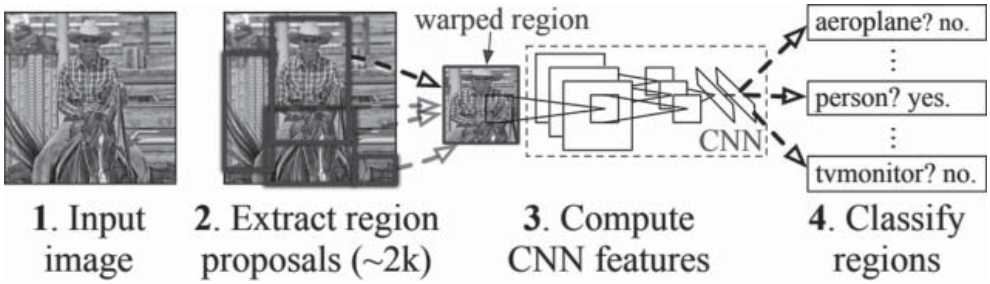


Рис. 6. Схема роботи R-CNN

тити сигнал далі. Мережа також використовує batch normalization. Наявність обхідних шарів дозволила натренувати модель зі 152 шарів, при цьому зберігаючи меншу складність ніж *VGGNet*.

ResNet після цього зазнала ще декількох модифікацій. В [17] запропоновано *Inception-v*4 (побудована на базі *GoogLeNet*, без обхідних з'єднань) та *Inception-ResNet* (містить обхідні з'єднання). Тут надано емпіричне доведення, що тренування з обхідними шарами пришвидшує тренування мереж з *Inception* архітектурою. Наведено деякі аргументи, що *ResNet Inception* архітектури (тобто поєднання *Inception* з обхідними з'єднаннями), не сильно, але все ж перевершують схожі за складністю Inception архітектури. Поєднання трьох *Inception-ResNets* та *Inception-v*4 дозволяє досягнути результату в 3.58% *top* 5 *error*.

В [18] викладена теорія про те, що *Residual Networks* поводять себе як ансамбль відносно неглибоких мереж. Автори переписують residual network як набір шляхів, які, як виявляється, ведуть себе як ансамбль, в тому сенсі, що вони не сильно залежать один від одного. Більшість градієнтів в мережі зі 110 шарів дають шляхи, які мають глибину всього 10—34.

Подальші покращення цієї архітектури зроблені в [19]. Взявши за основу інтерпретацію дуже глибоких residual networks як ансамблю мереж, автори будують мережу, яка більше широка, ніж глибока. Така архітектура не тільки покращує точність, але й дає значний приріст в швидкодії в тому сенсі, що її можна обчислювати паралельно на різних процесорах. Ідея широких мереж використана в [20], де показано, що навіть мережа глибиною в 16 шарів перевершує за швидкодією і точністю інші глибокі residual networks та розроблено архітектуру, яка показала найкращий результат на *CIFAR*-10.

В [21] представлено архітектуру (*FractalNet*), яка по суті є усіченим фракталом та генерує глибокі мережі, використовуючи просте правило розширення без застосування обхідних шарів, проте з арифметикою між шарами. Показано, що залишкові з'єднання не обов'язково є ключем до успіху, скоріше — це можливість перейти з неглибокого рівня на глибокий під час тренування.

SqueezeNet [22] — мережа, оптимізована під розмір моделі, містить в 50 разів менше зв'язків та є в 2 рази швидшою, ніж *AlexNet*, але має більшу точність. Після застосування компресії модель займає всього 470 KB.

Ідентифікація об'єктів. На відміну від класифікації зображень, ідентифікація об'єктів потребує локалізації об'єкта на зображенні.

Більшість робіт в цій області до 2014 р. базувалась на використанні *SIFT* (*scale-invariant feature transform* — перетворення ознак незалежно від масштабу) та *HOG* (*histogram of oriented gradient*s — гістограми орієнтованих градієнтів).

Одним з підходів до виявлення об'єктів є регресія, проте ця стратегія дала 30.5% *mAP* (*mean average precision*) на базі *VOC* 2007.

Одною з перших робіт, де використовувались згорткові нейронні мережі для виявлення об'єктів була [23], в якій представлено архітектуру *R-CNN* (*region-based CNN*). Автори використовують підхід розпізнавання

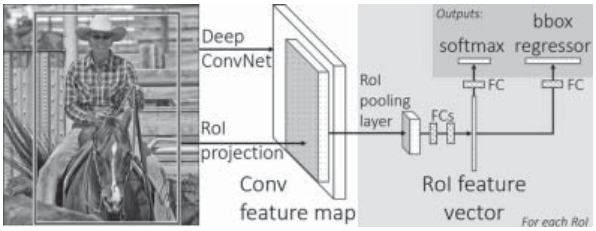


Рис. 7. Архітектура Fast R-CNN

на основі областей. Система складається з 3 модулів:

1) генерація областей-кандидатів (*RoI — region of interest, region proposals*), незалежних від категорій;

2) велика згорткова нейронна мережа для перетворення зображення у карту ознак (4096-вимірний вектор для кожної з областей кандидатів, згенерований використовуючи *AlexNet*);

3) набір специфічних для кожного класу лінійних *SVM*-класифікаторів.

Схему роботи *R-CNN* [23] представлено на рис. 6.

Потрібно зазначити, що цей метод дав значний приріст в точності в порівнянні з попередніми (53—58% *mAP* на *VOC* 2007-2012).

Цей підхід є досить складним в обчисленні, потрібно багато областей, що приводить до обчислень, які повторюються (для областей, що перетинаються). Цю проблему вирішує метод *Fast R-CNN* [24], шляхом пропускання цілого зображення через мережу для генерації карти ознак. Тоді для кожної області-кандидата агрегаційний шар бере вектор ознак фіксованого розміру з карти ознак. Цей вектор пропускається через декілька повнозв'язних шарів, які потім відгалужуються в два паралельних шари, один з яких генерує імовірності того, що область належить до певного класу, а інший (*bbox regressor*) генерує 4 значення, які описують рамку для об'єкта. Метод дав також приріст в точності — 68-70% *mAP* на *VOC* 2007-2012. Схему архітектури *Fast R-CNN* [24] наведено на рис. 7.

Fast R-CNN майже досягає швидкодії реального часу, якщо не брати до уваги час на генерацію областей-кандидатів (цей крок все ще дуже повільний). Слабким місцем є саме генерація областей. Запропоновано покращення цього алгоритму — *Faster R-CNN* [25]. Введено *RPN* (*region proposal network*), яка використовує ті ж згорткові ознаки цілого зображення, що і мережа виявлення об'єктів, таким чином уможливлюючи майже незатратну генерацію областей кандидатів. На першій стадії (*RPN*) зображення пропускаються через мережу *VGG*-16, яка генерує карту ознак, деякі з яких на певному внутрішньому шарі використовуються для генерації областей-кандидатів незалежно від класів. На другій стадії ці області (зазвичай 300) використовуються для вибору ознак з тієї

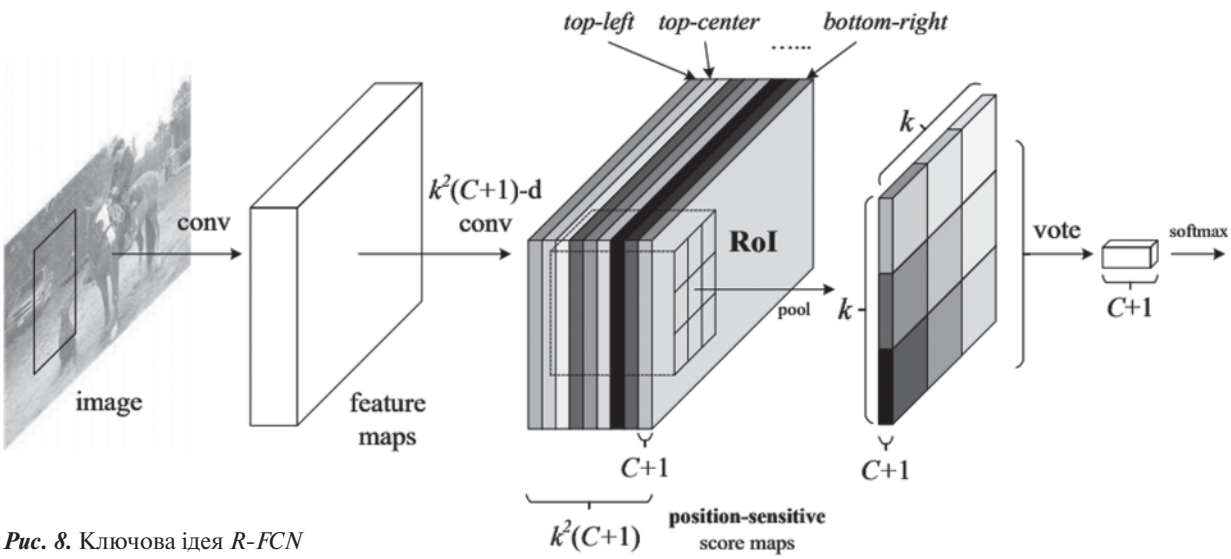
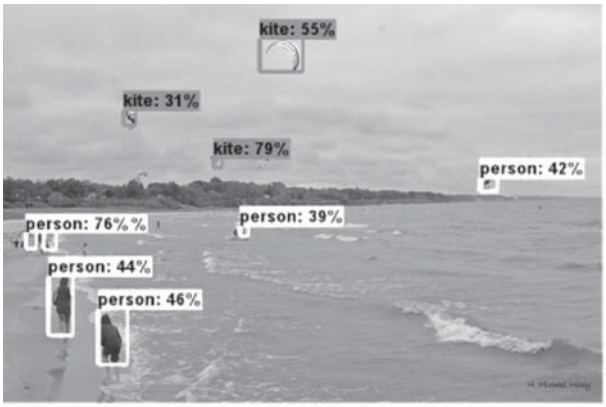
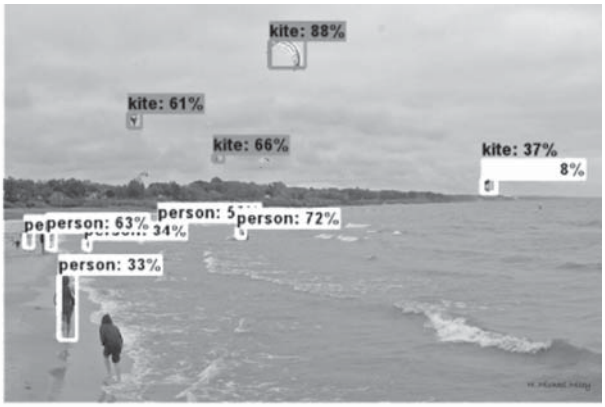


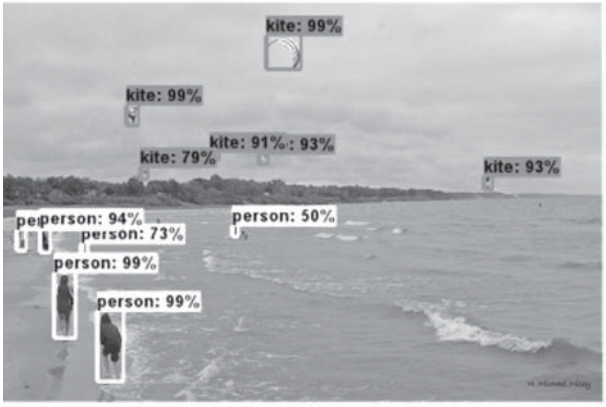
Рис. 8. Ключова ідея *R-FCN*



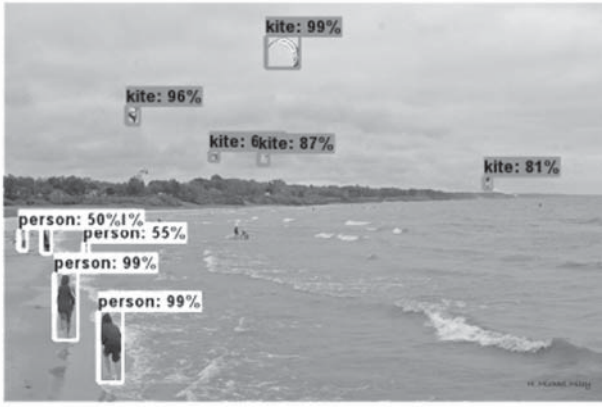
(a) SSD+Mobilenet, lowres



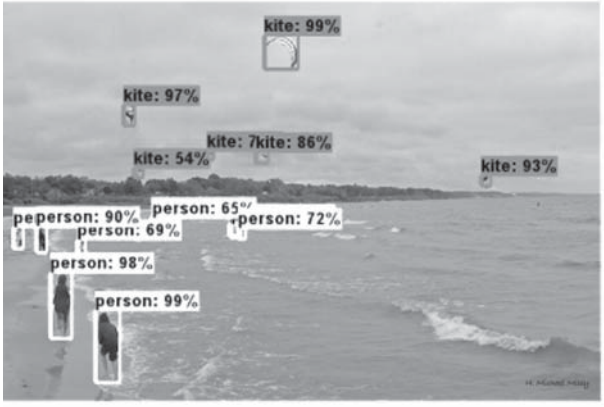
(b) SSD+InceptionV2, lowres



(c) FRCNN+Resnet101, 100 proposals



(d) RFCN+Resnet10, 300 proposals



(e) FRCNN+IncResnetV2, 300 proposals

Рис. 9. Результат роботи різних архітектур

ж карти, які після подаються на вхід останньої частини мережі генерації ознак з метою передбачення класу та уточнення обмежуючої рамки відповідно до класу. Області не генеруються напряму із зображення і не пропускаються кожна окремо через мережу генерації ознак, таким чином обчислення не повторюються для кожної області. Точність методу на *VOC* 2007-2012 складає 75—78% *mAP*. Архітектура дозволяє обробляти зображення зі швидкістю 5 fps.

Хоча *Faster R-CNN* на декілька порядків швидша за *Fast R-CNN*, той факт, що деякі об-

числення виконуються по кількасот разів для кожної області, привів авторів [26] до створення нового методу — *R-FCN*, який схожий з *Faster R-CNN*, проте мінімізує кількість обчислень, що припадають на кожну область. Замість того, щоб витягувати ознаки для області-кандидата напряму з карти ознак, вони переміщують цей процес на останній шар перед прогнозуванням. Всі шари є згортковими і обраховуються на цілому зображенні, а останній згортковий шар генерує k^2 чутливих до позицій карт оцінок для кожної категорії, і таким чином має $(C+1)$-канальний вихідний шар (C категорій + 1 для фону). Чутливість до позицій важлива в задачі виявлення об'єктів, на відміну від класифікації зображень. Після цього шару йде шар агрегації, який обирає з кожних $k \times k$ оцінок одну. Ключова ідея *R-FCN* зображена на рис. 8. Під час тренування, останній шар вчиться генерувати потрібні оцінкові карти. Показується, що *R-FCN* з використанням *ResNet* 101 (хоча можна застосовувати і інші мережі для класифікації) досягає точності такого ж порядку, як і *Faster R-CNN*, працюючи набагато швидше. Загалом запропонована архітектура працює в 2.5—20 разів швидше за попередника.

YOLO (*You Only Look Once*) [27] — це метод, який на відміну від інших, що використовують класифікатори в якості детекторів, розглядає проблему розпізнавання об'єктів як задачу регресії на просторово відокремлені обмежувальні прямокутники та пов'язані з ними класові імовірності. Одна нейронна мережа передбачає обмежувальні прямокутники та класові ймовірності з повного зображення за один прохід і працює з повним зображенням, а не з окремими областями. Архітектура має 24 згорткових і 2 повнозв'язних шари. Оскільки весь цикл розпізнавання є однією мережею, то його швидкодія може бути оптимізована. Архітектура є надзвичайно швидкою, але не настільки точною, як *Faster R-CNN*. Стандартна модифікація досягає 63.4% *mAP* при 45 *fps*, а швидка модифікація досягає 52.7% *mAP* при 155 *fps* на *VOC* 2007.

SSD (*Single Shot MultiBox Detector*) [28] — значний крок вперед в напрямку виявлення об'єктів в реальному часі. Автори запропонували метод, який досягає швидкості в 59 *fps* на *VOC* 2007 при цьому показуючи *mAP* 74.3%. Метод, як і попередній, не використовує генерацію областей-кандидатів. Натомість використовується невеликий згортковий фільтр для передбачення категорій і розміщення рамки та окремі фільтри для різного співвідношення сторін рамок. Ці фільтри застосовуються на різних етапах мережі з метою виявлення об'єктів різного масштабу.

На даний момент відомо дуже багато різних архітектур для виявлення об'єктів. Порівняльна статистика різних архітектур наводиться в [29]. Тут обрано архітектуру з оптимальним балансом швидкодії, пам'яті та точності, береться до уваги розмір об'єктів, що розпізнаються, та розширення зображення, що є дуже важливим в багатьох практичних задачах, зокрема в розпізнаванні повітряних об'єктів, оскільки вони зазвичай носять нечіткий характер та відносно малі на зображеннях, бо рухаються дуже швидко, а тому повинні бути виявленні на ранніх стадіях, коли об'єкт знаходиться ще досить далеко. Автори поєднують різні, як вони називають, мета-архітектури (*Faster R-CNN, R-FCN, SSD*) з різними мережами для класифікації (*VGG, Inception, ResNets* та ін.). Вони показують, що швидкодію *Faster R-CNN* можна покращити за рахунок використання меншої кількості областей-кандидатів без значних втрат в точності. Порівняння результату роботи різних архітектур наведено на рис. 9.

Складність нейронних мереж для класифікації зображень останніми роками зростала експоненційно, тому виникли роботи, присвячені автоматизації пошуку архітектури, де використовують рекурентні нейронні мережі і навчання з підкріпленням для пошуку потрібної архітектури.

Використання детекторів та дескрипторів в задачах комп'ютерного бачення

Значний прогрес досягнутий в області комп'ютерного зору завдяки використанню нейронних мереж. Багато IT гігантів працюють над розв'язанням задач комп'ютерного зору і біль-

шість з них роблять ставку саме на нейронні мережі. Потрібно зазначити, що хоча це один із найбільш успішних на даний момент підходів, він потребує великих затрат ресурсів: людської праці, фінансів, обчислювальних потужностей і т. н. Зрозуміло, що такі затрати доступні далеко не кожному, саме тому значного прогресу в індустрії досягли *Google, Facebook* та інші великі корпорації.

Проте існують методи, які дозволяють отримати бажаний результат при значно менших затратах, хоча і у задачах вузького спрямування, а комбінацією таких методів можна досягнути розв'язання більш широкої задачі. На відміну від нейронних мереж, які фокусуються на моделюванні аналітичних підходів людини, ці методи базуються на особливостях самого зображення. Одними із таких є підходи, що базуються на знаходженні так званих особливих точок та їх числового представлення. Такі методи дозволяють комп'ютеру досить точно працювати з візуальною інформацією.

Щоб описати зображення, потрібно прив'язатись до його локальних особливостей або як їх часто називають — особливих точок. Особлива точка — це така точка зображення, що задовольняє ряду властивостей:

1. визначеності — особливість повинна виділятися на фоні серед сусідніх точок;

2. стійкості — зміна яскравості, контрастності і кольорової гами не повинні впливати на місце особливої точки на об'єкті або сцені;

3. інваріантності — особливі точки повинні володіти стійкістю до повороту, зміни масштабу зображення і зміни ракурсу зйомки;

4. стабільності — зашумленість зображення, що не перевищує певний поріг, не повинна впливати на роботу детектора;

5. інтерпретованості — особливі точки повинні бути представлені в форматі, придатному для подальшої роботи;

6. кількості — кількість виявлених особливих точок має відповідати мінімальній вимозі до їх кількості для зіставлення зображень.

Пошук особливих точок здійснюється детектором. Детектор — алгоритм пошуку особливих точок.

Дескриптор — опис особливої точки, що визначає особливості її околиці, являє собою числовий або бінарний вектор певних параметрів. Довжина вектора і вид параметрів визначаються застосовуваним алгоритмом. Дескриптор дозволяє виділити особливу точку з усієї їх множини на зображенні. Це необхідно для складання ключових пар особливостей, що належать одному об'єкту при порівнянні різних зображень.

Схема застосування детекторів і дескрипторів для розв'язання задачі класифікації. Одними з найпоширеніших класів алгоритмів класифікації є так звані *bag-of-words* (або *bag-of-features, bag-of-key-points*). Ідея запозичена з задачі класифікації текстів, де використовується опис у вигляді гістограм входжень певних слів із наперед складеного словника. Основні кроки таких алгоритмів описуються так:

1. виявлення ключових точок зображення;

2. обчислення дескрипторів локальних околів особливих точок;

3. кластеризація дескрипторів ключових точок, що належать всім об'єктам навчальної вибірки;

4. побудова опису кожного зображення у вигляді нормованої гістограми входжень «слів» (для кожного кластеру обчислюється кількість віднесених до нього ключових точок певного зображення);

5. побудова класифікатора, який використовує обчислений на кроці 4 опис зображення;

Дескриптори, що використовуються алгоритмами даного класу, повинні бути інваріантні до афінних перетворень та зміни освітлення. Словник дескрипторів ключових точок повинен бути достатньо великим, щоб відображати релевантні зміни частин зображення, але в той же час не надто великим, щоб алгоритм був стійким до шуму.

Такий підхід отримав досить критики, оскільки він ніяк не враховує просторову інформацію про розподіл ключових точок зображення в результаті чого опис об'єктів зі схожими наборами ключових точок, які знаходяться в абсолютно різних конфігураціях, співпадає. Для врахування цього аспекту запропоновано

декілька підходів: використання так званих просторових корелограм «візуальних слів»; використання ієрархічної моделі, в якій об'єкт представляється P частинами, до кожної з яких відноситься N_p ключових точок; використання схеми порівняння пірамід, в якій простір ознак розбивається на послідовність вкладених одна в одну підобластей і обчислюється зважена сума числа співпадінь на всіх рівнях розбиття, при чому співпадіння на більш детальних рівнях мають більшу вагу.

Ще одним підходом до розв'язання задачі класифікації є використання моделей об'єктів, що складаються з частин (*part-based models*). Алгоритми даного класу враховують взаємне розташування різних частин об'єкту. Наприклад, при розпізнаванні обличчя важливе взаємне розташування очей, носа, рота, волосся тощо. Основними елементами моделей, що складаються з частин є:

- представлення окремих частин об'єкта (зазвичай для цього використовують дескриптори);
- методи навчання даного представлення;
- опис зв'язків між частинами.

Виявлення об'єктів за допомогою детекторів і дескрипторів. Методи розв'язання задач виявлення об'єктів можна умовно поділити на 3 групи:

- методи, що використовують найбільш характерні ознаки для опису об'єктів. В якості ознак можуть бути вибрані точкові особливості об'єкта або ознаки побудовані для зображення, що містить тільки один цей об'єкт;
- методи пошуку об'єктів за шаблоном;
- методи виявлення об'єктів, що рухаються на базі декількох зображень або кадрів відео однієї й тої ж сцени.

Методи на основі характерних ознак спочатку будують характерні вектори ознак для особливих точок об'єкта або для всього об'єкта, а потім на основі цього будують класифікатор (для побудови класифікатора можуть використовуватись також методи машинного навчання). Оскільки об'єктів на зображенні може бути досить багато і вони можуть бути представлені в різних масштабах, то потрібно проглядати області зображення використовуючи техніку «ковзаючої рамки» різного розміру. Детектори і дескриптори використовуються для виявлення та опису особливих точок. Ця інформація потім подається на вхід класифікатора, який відносить об'єкт до певного класу.

Методи пошуку за шаблоном в якості шаблону можуть використовувати зображення, на якому присутній тільки шуканий об'єкт, або дескриптори, які притаманні цьому об'єкту. Для розв'язання задачі як і в інших методах застосовується техніка «ковзаючої рамки» різних масштабів. Частина зображення, що попадає в рамку співставляється з шаблоном. Результатом такого порівняння є міра схожості, в якості якої може бути вибрана, наприклад, величина, обернена до Евклідової відстані. Вважається, що, якщо значення схожості більше якогось наперед вибраного порогу, то знайдено співпадіння. Проблема вибору порогу є важливою, оскільки його збільшення призводить до збільшення числа співпадінь, які були хибно відкинуті, тоді як зменшення порогу призводить до збільшення числа хибних співпадінь. Методи пошуку за шаблоном ефективно працюють при пошуку одиночних об'єктів, тоді як за наявністю багатьох об'єктів різних класів на зображенні виникають певні труднощі, пов'язані з перекриттям об'єктами один одного, що призводить до відсутності певних ознак в результуючих дескрипторах.

Порівняння зображень часто відбувається у три кроки: виділення двох множин так званих особливих точок; побудова дескрипторів особливих точок; зіставлення особливих точок.

Цікавий підхід на основі співставлення зображень використовується в [30]. Автори використовують особливі точки, на основі яких шукають відстані між вихідним та еталонним зображеннями для розпізнавання відбитків пальців.

Алгоритми порівняння дескрипторів. Найпростішим алгоритмом порівняння є brute force або метод повного перебору, тобто всі дескриптори першого зображення зіставляються з усіма дескрипторами другого зображення. Збігом вважається дескриптор, відстань до якого найменша. Відстань обирається в

залежності від розглядуваної задачі та вигляду дескрипторів.

Пошук відповідного дескриптора — це пошук найближчого сусіда в просторі дескрипторів. Бібліотека *FLANN* (*Fast Library for Approximate Nearest Neighbors*) містить колекцію алгоритмів для пошуку найближчих сусідів в просторах великої розмірності. Методи бібліотеки автоматично вибирають найкращий алгоритм пошуку і оптимальні параметри, базуючись на вхідних даних. В проведених розробниками експериментах, реалізовані в бібліотеці алгоритми пошуку, на порядок швидші в більшості відомих баз даних, ніж їх попередники.

Відфільтрувати дескриптори тільки за відстанню недостатньо для досягнення високої точності визначення схожих об'єктів на зображеннях. Якщо об'єкт рухається на сцені або знятий з іншого ракурсу, то при застосуванні викладених вище методів порівняння дескрипторів можуть з'явитися помилково визначенні зв'язки.

Для зниження впливу помилкових зіставлень особливих точок використовується алгоритм робастного оцінювання *RANSAC* (*RANdom Sample Consensus*) [31]. Під робастністю слід розуміти нечутливість до малих відхилень від припущень. Цей допоміжний алгоритм, використовується як варіант додаткової фільтрації результатів, відсіювання хибних співпадінь після знаходження збігів особливих точок.

RANSAC — це метод, який використовується для оцінки параметрів моделі на підставі випадкових вибірок. При зіставленні модель являє собою матрицю перетворення (гомографія). На вході алгоритму є дві множини дескрипторів, отриманих на попередньому і поточному зображенні.

Схема роботи *RANSAC* полягає в багаторазовому повторенні таких етапів.

1. Вибір опорних точок і побудова параметрів моделі, що містить 8 невідомих параметрів. Для пошуку матриці гомографії необхідно, як мінімум, 4 пари особливих точок на зображеннях, що зіставляються. На підставі отриманих наборів будується матриця перетворення.

2. Перевірка побудованої моделі. Для кожної точки попереднього кадру знаходиться проекція на поточному кадрі, і виконується пошук найбільш близького дескриптора (точки) з множини дескрипторів поточного кадру. Характерна точка позначається як «викид», якщо відстань між проекцією і відповідним дескриптором поточного зображення більше заданого порогу.

3. Заміщення моделі. Перевіряється, чи є побудована модель кращою серед набору попередніх моделей.

В результаті застосування *RANSAC* будується найкраща матриця гомографії, яка буде використовуватися для пошуку помилкових відповідностей.

В даному випадку алгоритм *RANSAC* — це математичний засіб верифікації достовірності виявлених збігів.

Детектори.

Детектор Моравеця. Одним з найперших і найпростіших детекторів кутів є алгоритм Моравеця [32]. Автор розглядає зміну яскравості квадратного вікна (зазвичай розміру 3×3, 5×5, 7×7) відносно розглядуваної точки при зсуві вікна на 1 піксель у 8 напрямках (горизонтальні, вертикальні, діагональні). Для кожного пікселя обчислюється зміна інтенсивності, після чого будується карта ймовірності знаходження кутів в кожному пікселі за допомогою оціночної функції. Потім пікселі зі значенням ймовірності меншим певного порогового значення відкидаються і в кінці, за допомогою процедури пошуку локальних максимумів функції відгуку, видаляються кути, що повторюються. В результаті всі ненульові елементи, що залишились, відповідають кутам. Серед недоліків методу варто відзначити такі: неінваріантність до повороту та помилки виявлення при великій кількості діагональних ребер.

Детектор Кенні. Детектор був запропонований Джоном Кенні в 1986 [33]. Алгоритм складається з 4 кроків:

1. до зображення застосовується фільтр Гаусса з маскою *K*. Відбувається розмиття зображення, тим самим видаляється той шум, що залишився;

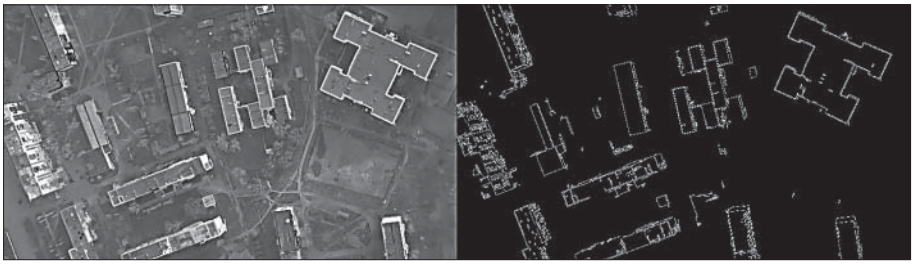


Рис. 10. Приклад роботи детектора кутів Кенні

2. пошук максимальних значень градієнтів. Для кожного пікселя зображення застосовується оператор Собеля, який обчислює наближене значення градієнта яскравості зображення. Далі обчислюється напрямок градієнта;

3. пригнічення не-максимумів. Пікселями контурів є пікселі, в яких досягається локальний максимум градієнта в напрямку вектора градієнта. Значення напрямку повинно бути кратне 45°;

4. порогова фільтрація для визначення потенційних границь. Застосовується для визначення, чи знаходиться границя в даній точці зображення. Чим менший поріг тим більше границь знаходиться, але тим більш сприйнятливим до шуму стане результат, виділяючи зайві дані зображення. Високий поріг може проігнорувати слабкі межі або отримати границю фрагментами. Виділення границь Кенні використовує два порога фільтрації: якщо значення пікселя вище верхньої межі — він приймає максимальне значення (межа вважається достовірною), якщо нижче — піксель пригнічується. Точки із значенням, що потрапляють в діапазон між порогів, приймають фіксоване середнє значення. Результат роботи детектора кутів Кенні наведено на рис. 10.

Детектор Харріса і Стівенса. Харріс і Стівенс [34] покращили детектор Моравеця розглядаючи похідні за деякими напрямками безпосередньо, а не використовуючи зсув рамки.

Далі на основі власних чисел матриці Харріса — робиться висновок про характер точки:

1) якщо обидва власні числа досить великі — кут;

2) якщо одне власне число значно більше іншого — ребро;

3) якщо обидва власні числа близькі до нуля, то поточний піксель належить плоскій області.

Серед недоліків детектора Харріса і Стівенса: більші обчислювальні затрати порівняно з детектором Моравеця, чутливість до шуму. Серед переваг: інваріантність відносно повороту і менша кількість помилок виявлення кутів.

FAST (Features from Accelerated Segment Test). В алгоритмі розглядається коло з 16 пікселів (побудоване алгоритмом Брезенхема) навколо точки-кандидата *P*. Точка є кутовою, якщо для поточної розглянутої точки *P* існують *N* суміжних пікселів на колі, інтенсивність яких більше $I_P + t$ або інтенсивності всіх менше $I_P - t$, де I_P — інтенсивність точки *P*, *t* — гранична величина. Далі необхідно порівняти інтенсивність в вертикальних і горизонтальних точках на колі під номерами 1, 5, 9 і 13 з інтенсивністю в точці P (для того, щоб якомога швидше відстежити хибні кандидати). Якщо для трьох з цих точок виконається умова $I_{P_i} > I_P + t$ або $I_{P_i} < I_P + t$, $i = 1, ..., 4$ то проводиться повне тестування для всіх 16 точок [35]. Експерименти показали, що найменше значення *N*, при якому особливі точки починають стабільно проявлятися: $N = 9$.

Існують різні модифікації алгоритму: древовидні *FAST*-9 і *FAST*-12.

Хоча алгоритм є одним із найпоширеніших на даний момент, він все ж має ряд недоліків: в околі деякої точки може бути виявлено декілька особливих точок, ефективність алгоритму залежить від порядку обробки зображення та розподілу пікселів.

В [36] покращено алгоритм використовуючи машинне навчання. Покращена версія отримала назву *FAST-ER* (*ER — Enhanced Repeatability*, покращена повторюваність). На одній і тій

же сцені, що розглядається з різних ракурсів, алгоритм знаходить особливі точки, що належать одним і тим же об'єктам. *FAST-ER* в загальному краще виконує поставлене завдання, ніж *FAST*, але він потребує більших обчислювальних ресурсів.

В [37] автори запропонували альтернативу *FAST — AGAST* (*Adaptive and Generic Accelerated Segment Test*). Автори використовують той же *AST*, але покращують побудову і використання дерев рішень для *AST*. Алгоритм застосовує метод обчислення двійкового дерева рішень (кутового детектора), який є загальним і не повинен бути адаптований до нових середовищ. Комбінуючи два дерева, кутовий детектор автоматично адаптується до навколишнього середовища і забезпечує найбільш ефективне дерево рішень для області зображення із затримкою в один піксель. *AGAST* використовує таке ж пригнічення немаксимумів, що і *FAST*.

Дескриптори. На вхід дескрипторам подається зображення і набір особливих точок, а виходом є набір векторів ознак для кожної особливої точки. Проте деякі алгоритми розв'язують одразу дві задачі: пошук особливих точок та побудову їх дескрипторів.

Дескриптори для отримання якісного результату повинні мати такі властивості:

1) повторюваність — більшість ознак повинно зберігатись при зміні точки зйомки чи умов освітлення;

2) локальність — ознаки повинні якомога більш локальними, щоб уникати перекриття одних точок іншими;

3) репрезентативність — кількість ознак повинна бути достатньою для того, щоб розумне число ознак виявлялось навіть на невеликому зображенні;

4) ефективність — економія обчислювальних ресурсів, що особливо важливо в задачах реального часу.

SIFT (Scale Invariant Feature Transform) [38]. Алгоритм можна описати наступним чином.

1. Побудова простору зображень, що масштабується — набору зображень, згладженим Гаусовим фільтром.

2. На основі отриманого простору зображень, що масштабується, обчислюється різниця Гауса — попіксельне віднімання зображень в одній октаві. Октаву формують зображення одного масштабу, розмиті фільтром Гауса з різним радіусом розмиття (4 зображення в одній октаві). На цьому етапі забезпечується інва-

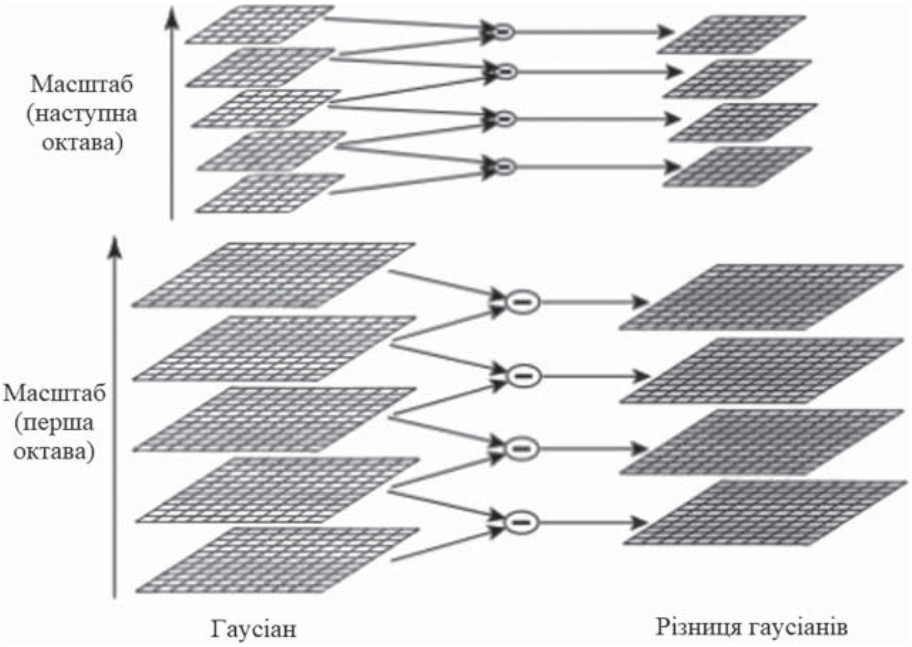


Рис. 11. Різниця Гаусово-розмитих зображень

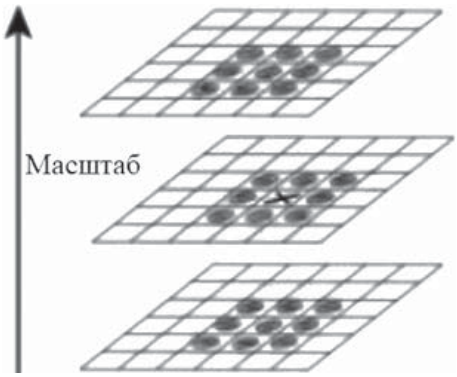


Рис. 12. Схема методу пошуку екстремумів різниці гаусіанів

ріантність до масштабування. Потім визначаються екстремуми, які записуються в список потенційних особливих точок.

На рис. 11 зліва зображена піраміда гаусіанів, а справа — їх різниць. Кожна різниця виходить з двох сусідніх гаусіанів. При переході до наступної октави розмір зображень зменшується вдвічі. Після побудови пірамід точка вважається особливою, якщо вона є локальним екстремумом різниці гаусіанів. Для пошуку екстремумів використовується метод, схематично зображений на рис. 12. Якщо значення різниці гаусіанів в точці, позначеній хрестиком більше (менше) всіх значень в точках, позначених колами, то ця точка вважається точкою екстремуму.

3. Інваріантність до повороту досягається шляхом обчислення гістограми орієнтацій для деякої ключової точки. Для кожного значення метод знаходить екстремальні значення гістограми орієнтації.

Після визначення орієнтації всі операції виконуються на зображеннях, перетворених відносно призначеної орієнтації, масштабу і розташування кожної характеристики, чим забезпечується інваріантність цих перетворень.

4. Обчислення дескрипторів. Дескриптор — вектор довжиною $128 = 8$ (кількість бінів) $\times 4 \times 4$ (кількість квадратів). В кожному квадраті розміру 4×4 обчислюється гістограма орієнтованих градієнтів шляхом додавання зваженого значення магнітуди градієнту до одного з 8 бінів гістограми. Використовується також так звана білінійна інтерполяція для уникнення віднесення схожих градієнтів до різних квадратів: значення магнітуди кожного градієнта додається не тільки в гістограму відповідного квадрату, але і в гістограми сусідніх квадратів з вагою пропорційною відстані від пікселя, в якому градієнт обчислений до центру відповідного квадрату.

Перевагами алгоритму є його інваріантність до поворотів, масштабу, зсувів та частково до зміни освітлення. Недоліком є вимогливість до обчислювальної потужності. Крім того, комерційне використання алгоритму *SIFT* повинне бути ліцензоване.

Алгоритм зазнав декілька модифікацій, серед яких *PCA-SIFT* (*PCA — Principal Component Analysis*) та *GLOH* (*Gradient Location and Orientation Histogram*).

SURF (Speeded up Robust Features) [39]. Алгоритм розв'язує обидві задачі — задачу пошуку особливих точок та обчислення дескрипторів інваріантних до масштабу та повороту.

Метод шукає особливі точки за допомогою матриці Гессе. Гессіан досягає екстремуму в точках максимальної зміни градієнту яскравості. Алгоритм добре виявляє плями, кути, краї. Оскільки гессіан не інваріантний відносно масштабу, то *SURF* використовує різномасштабні фільтри для знаходження гессіанів. Для кожної ключової точки обчислюється напрямок максимальної зміни яскравості і масштаб, взятий з масштабного коефіцієнта матриці Гессе. Градієнт обчислюється за допомогою фільтрів Хаара.

Після знаходження ключових точок формуються дескриптори — набір з 64 (або 128) чисел. Ці числа відображають напрямок градієнта навколо ключової точки. Оскільки ключова точка являє собою максимум гессіана, то це гарантує те, що в оточенні точки повинні бути ділянки із різними градієнтами. Таким чином забезпечується дисперсія (відмінність) дескрипторів для різних ключових точок. Напрямок градієнта околу рахується відносно напрямку градієнта навколо точки в цілому (по всьому околу ключової точки). Таким чином, досягається ін-

варіантність дескриптора відносно обертання. Розмір області, на якій рахується дескриптор, визначається масштабом матриці Гессе, що забезпечує інваріантність відносно масштабу. Напрямок градієнта також рахується за допомогою фільтра Хаара.

Для ефективного обчислення фільтрів Гессе і Хаара використовується інтегральне представлення зображення.

Для обчислення дескрипторів виконуються наступні кроки:

- навколо області будується квадратний окіл розміром $20s$, де s — масштаб, на якому отримано максимум гессіана;
- отримана квадратна область розбивається на регіони 4×4;
- для кожного блоку обчислюються прості ознаки, в результаті чого отримується вектор з чотирьох компонент: дві — сумарний градієнт за квадрантом, дві — сума модулів точкових градієнтів;
- дескриптор формується в результаті склеювання зважених описів градієнту для шістнадцяти квадрантів навколо особливої точки. В якості ваг використовуються коефіцієнти Гаусівського ядра, які необхідні для стійкості до шумів у віддалених точках;
- до дескриптора додається слід матриці Гессе для розрізнення темних (слід додатній) і світлих (від'ємний) плям.

Алгоритм дав значний приріст у швидкості у порівнянні зі своїми попередниками, що автори вважають найбільшою заслугою цього алгоритму. За показник якості автори вибрали повторюваність, яку вони вимірювали роблячи різного роду трансформації з вхідним зображенням.

Перевагами алгоритму є інваріантність відносно повороту та масштабування, інваріантність відносно різниці загальної яскравості, може виявляти багато об'єктів на сцені. Недоліками є складність реалізації, відносно повільна робота алгоритму.

BRIEF (Binary Robust Independent Elementary Features). Алгоритм представлено в [40] з метою розпізнавання однакових областей зображення, що зняті з різних точок. За мету автори ставили максимальне зменшення кількості обчислень. Алгоритм зводиться до побудови випадкового лісу або наївного Баєсівського класифікатора на тренувальній множині. Для найбільш схожої області в тренувальній вибірці може використовуватись, наприклад, метод найближчого сусіда. Невеликої кількості операцій вдалось добитись за рахунок представлення вектора ознак у вигляді бінарного рядка і використання відстані Хеммінга в якості міри близькості.

Зображення розбивається на області розміром, скажімо, $S \times S$. З області вибирається множина p пар пікселів (x, y), де x та y вектори вигляду $(u, v)^T$. Для цієї множини будується набір бінарних тестів:

$$\tau(p; x, y) = \begin{cases} 1 \ if \ p(x) < p(y) \\ 0 \ otherwise \end{cases},$$

де $p(x)$ — інтенсивність пікселя x в згладженій версії області p.

Вибір *множини* n_d *пар* ідентифікує множину бінарних тестів і *BRIEF* дескриптор будується наступним чином:

$$f_{n_d} = \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_i).$$

Автори *беруть* n_d *рівним* 128, 256, 512. В експериментах автори беруть точки згідно з рівномірним та нормальним розподілами з різними значеннями математичного сподівання та середньоквадратичного відхилення. Алгоритм дає результати порівнювані з *SURF*, а на деяких тестових даних навіть перевершує *SURF* та *SIFT* алгоритми. Алгоритм виконується значно швидше, за своїх попередників, проте він є досить чутливим до значних поворотів.

BRISK (Binary Robust Invariant Scalable Keypoints). У [41] представлено метод *BRISK*, який є розвитком *SURF* у плані подальшого удосконалення складових *FAST*.

Враховуючи розташування ключових точок та відповідні значення масштабу, дескриптор *BRISK* складає дескриптор — двійковий рядок шляхом об'єднання результатів порівняльних тестів яскравості. Ідентифікується характерний напрямок кожної ключової точки, щоб отримати орієнтовано нормалізовані

дескриптори та забезпечити інваріантність до обертання.

Концепція дескриптора *BRISK* використовує шаблон шляхом аналізу точок, що розташовані рівномірно розподілено по колам, концентричним з ключовою точкою. Це забезпечує інтегрований аналіз та високу швидкість оброблення чи зберігання. *BRIEF*-дескриптор тут забезпечує розпізнавання однакових ділянок зображення, знятих з різних точок зору.

У *BRISK* порівняно з *SIFT* та *SURF* приблизно однакова точність розпізнання зображень, але при цьому досягнута в декілька разів вища швидкодія реалізації. Відзначимо, що на окремих тестових зображеннях точність детектування за допомогою *BRISK* значно вище, ніж з використанням *SURF* дескрипторів. Автори також зазначають, що алгоритм також добре пристосовується до різних модифікацій, так неважко збільшити швидкодію алгоритму за рахунок зменшення розглядуваних особливих точок або легко можна позбутись інваріантності до масштабування та/чи повороту у випадках, коли вони непотрібні.

Таким чином основні переваги *BRISK* полягають у забезпеченні суттєво вищої швидкодії за рахунок спрощення процесу оброблення та побудови і використання дескриптора бінарного типу.

ORB (Oriented FAST and Rotated BRIEF). Алгоритм, представлений в [42], є ефективною альтернативою *SIFT* та *SURF*. *ORB* є свого роду сплавом *FAST* детектора та дескриптора *BRIEF*. За допомогою детектора *FAST* знаходяться особливі точки, після чого застосовується міра Харріса для визначення найкращих *N* точок. Алгоритм використовує пірамідальну структуру для виявлення особливостей різних масштабів. Для обчислення дескриптора використовується *BRIEF*, проте він не є інваріантним до значних поворотів, тому автори роблять певну модифікацію у відповідності до напрямку особливої точки, для того, щоб мати можливість застосувати *BRIEF*. Для розрахунку дескриптора в околиці особливої точки на поточному масштабі виділяється область, для якої знаходиться центр мас. Вектор, спрямований з особливої точки в бік центру мас, буде задавати орієнтацію особливої точки. Для складання дескриптора необхідно сформувати квадратне вікно, центроване щодо ключової точки і узгоджене з її орієнтацією. У цьому вікні за заданим правилом вибирається набір пар точок, значення яскравості в яких порівнюються між собою. Якщо яскравість першої точки буде вищою, у відповідний елемент дескриптора записується 1, в іншому випадку записується значення 0. Складені таким чином дескриптори можна зіставляти один з одним за нормою Хемінга.

Структура алгоритму *ORB* показує, що він є менш вимогливим до обчислювальних ресурсів. Виграш в швидкості обчислень визначається, перш за все, більш простою процедурою побудови дескрипторів і механізмом обчислення норми.

KAZE. Алгоритм був представлений в 2012 р. [43] і залишається у відкритому доступі. Ідея створення цього алгоритму полягала у виявленні та описі 2D особливості в нелінійних екстремумах масштабного простору, щоб отримати кращу точність локалізації. Гаусове розмиття, що використовується в інших алгоритмах розпізнавання об'єктів, таких як *SIFT*, не враховує природні межі об'єктів, оскільки деталі зображення і шум згладжуються однаковою мірою на всіх рівнях шкали.

Щоб зробити розмиття пристосовним до особливостей зображення, *KAZE* використовує нелінійну диффузійну фільтрацію разом з методом *AOS* (*additive operator splitting* — адитивне розкладання операторів). При такій фільтрації шуми зображення зменшуються, але межі об'єктів зберігаються.

Для розпізнавання об'єктів *KAZE* слідує в основному тими ж кроками, що і *SIFT*, але з деякими відмінностями на кожному кроці.

Цей алгоритм використовує нелінійну диффузійну фільтрацію в поєднанні з функцією провідності замість гауссівского масштабного ядра, використовуваного в *SIFT*. Ця функція провідності приймає градієнт гауссової згладженої версії вихідного зображення як функцію часу. Ця діяльність спрямована на отри-

мання особливостей, які мають більш високу повторюваність і відмінність ніж *SIFT*.

Оскільки *KAZE* обчислює багатомасштабні похідні (градієнти) для кожного пікселя, алгоритм *KAZE* більш вимогливий до продуктивності, ніж *SURF*, але його можна порівняти з *SIFT*. *KAZE* економить обчислювальні затрати при описі ключових точок, тому що один і той же набір похідних використовується для опису ключової точки.

AKAZE (Accelerated KAZE). Пришвидшена версія *KAZE* представлена в [44]. Автори ставили за мету покращити швидкість роботи як детектора, так і дескриптора. При цьому знайдені особливі точки і їх дескриптори повинні задовольняти високим показникам точності при порівнянні зображень.

Застосування алгоритму *FED Fast Explicit Diffusion* — на пірамідальній схемі дозволяє побудувати нелінійну багатомасштабну піраміду. Використання нелінійного коефіцієнта масштабування дозволяє збільшити швидкість знаходження потрібної особливої точки в порівнянні з Гауссовою пірамідою. Обчислення даного коефіцієнта базується на зміні яскравості зображення при масштабуванні.

Порівняння різних детекторів та дескрипторів. В [45] автори використовують алгоритми *ORB* та *AKAZE* для порівняння візуальної одометрії отриманої з безпілотних літальних апаратів. Вони використовують дві різні бази даних знімків, зроблені з різних висот та різних камер. Отриману інформацію вони фільтрують за допомогою *RANSAC* і порівнюють з метою визначення положення безпілотника. Застосування алгоритмів у такого вигляду системах ставить додаткові жорсткі вимоги на обчислювальні затрати і швидкодію. Автори приходять до висновку, що *ORB* є швидшим в обчисленні та час виконання *AKAZE* швидко зростає зі збільшенням роздільної здатності. Проте, після фільтрування і видалення так званих викидів, виявляється, що *AKAZE* знаходить більше правильних збігів, ніж ORB. Хоча *ORB* є швидшим, *AKAZE* демонструє кращий компроміс між точністю та швидкістю на знімках з малим розширенням — 680×480. Автори приводять отримані знімки з високою роздільної здатністю до такого розміру з метою забезпечити обробку в режимі реального часу.

Схоже порівняння проводиться в [46], де автори порівнюють *SIFT*, *SURF*, *ORB*, *AKAZE* та *BRISK* для задачі реконструкції спостережень. Результати, що наводяться, підтверджують сказане вище щодо швидкодії алгоритмів. *ORB* та *BRISK* показують найкращу швидкість, але генерують дуже багато викидів, що потенційно збільшує час порівняння зображень. Тоді як *AKAZE* працює швидше, генерує набагато менше точок, проте лише невелика частина згенерованих результатів є викидами.

Порівняння різних комбінацій детекторів і дескрипторів зроблено в [47]. Розглянуто комбінації: *SIFT* і *SIFT*, *SURF* і *SURF*, *MSER* і *SIFT*, *BRISK* і *FREAK*, *BRISK* і *BRISK*, *ORB* і *ORB* та *FAST* і *BRIEF*. Багато різних метрик використовується для порівняння, основні з яких — точність (відношення правильних співпадінь до повної кількості співпадінь) та повнота (відношення правильних співпадінь до кількості відповідностей між оригінальним зображенням та отриманим знімком). Автори розглядають алгоритми в контексті різних аспектів: вплив масштабування та повороту (*ORB* показує найкращі результати в більшості випадків), вплив розмиття (знову ж таки *ORB* в більшості перевершує інші підходи, проте результати приблизно одного порядку показують *MSER* і *SIFT* та *FAST* і *BRIEF*), вплив зміни точки зйомки (найкращі *MSER* і *SIFT* та *ORB*), вплив зміни освітлення (*FAST* і *BRIEF* виділяється на фоні інших), вплив компресії *JPEG* (*ORB* перевершує інші підходи). Конкретні результати порівнянь наведено в [47]. Потрібно зазначити, що використання *ORB* є допустимим у всіх розглянутих випадках. Найшвидшою є комбінація *FAST* і *BRIEF*, далі йдуть *ORB* та *BRISK* з на порядок нижчими показниками. Для коректного порівняння швидкості виконання потрібно використовувати також інші бази даних та більше зображень.

Ґрунтовне порівняння детекторів та дескрипторів проведено в 2018 р. в [48]. Розглядаються недоліки всіх попередніх по-

рівнянь та використовуються дві бази даних, скомбіновані з різних зображень багатьох інших баз даних. Порівнюються алгоритми реалізовані в бібліотеці *OpenCV*: *SIFT*, *SURF* (128), *SURF* (64), *KAZE*, *AKAZE*, *ORB*, *ORB* (1000) — *ORB* з параметром *MaxFeatures* = 1000 (на відміну від звичайного, в якому цей параметр дорівнює 100000), *BRISK*, *BRISK* (1000) — *BRISK* з *MaxFeatures* = 1000. Автори порівнюють вплив різноманітних перетворень на роботу алгоритмів, а також звертають увагу на швидкодію роботи різних частин та алгоритму в цілому. Деталізоване порівняння можна знайти в [48], наведемо тільки деякі з отриманих результатів. Автори дійшли висновку, що *SIFT*, *SURF* та *BRISK* є найбільш інваріантними до зміни масштабу, в той час як *ORB* — найменш інваріантний до цієї зміни. *ORB* (1000), *BRISK* (1000) та *AKAZE* — більш інваріантні до повороту ніж інші. *ORB* та *BRISK* найбільш ефективні алгоритми, які виявляють також велику кількість особливостей, проте час потрібний на співставлення всіх цих особливостей може впливати на загальний час порівняння зображень. Тоді як *ORB* (1000) та *BRISK* (1000) мають найменший час порівняння, проте за рахунок цього втрачається точність. Загалом *SIFT* та *BRISK* найточніші, якщо розглядати всі геометричні перетворення в цілому, з перевагою на боці *SIFT*.

В [48] наводяться такі упорядкування алгоритмів:

- за вмінням виявляти велику кількість особливостей:

ORB>*BRISK*>*SURF*>*SIFT*>*AKAZE*>*KAZE*;

- за ефективністю обчислення однієї особливої точки:

ORB>*ORB*(1000)>*BRISK*>*BRISK*(1000)>>*SURF*(64)>*SURF*(128)>*AKAZE*>*SIFT*>*KAZE*;

- за середнім часом витраченим на співставлення особливостей однієї точки:

ORB(1000)>*BRISK*(1000)>*AKAZE*>*KAZE*>>*SURF*(64)>*ORB*>*BRISK*>*SIFT*>*SURF*(128);

- за загальним часом витраченим на співставлення:

ORB(1000)>*BRISK*(1000)>*AKAZE*>*KAZE*>>*SURF*(64)> *SIFT*>*ORB*>*BRISK* >*SURF*(128).

Загалом є дуже багато робіт, які порівнюють між собою різні алгоритми, проте вони використовують різні бази даних і різні критерії якості, розглядаються в контексті розв'язання різних задач. Тому неможливо зробити однозначне порівняння алгоритмів і розставити їх в порядку якості. Порівняння повинне проводитись в залежності від розв'язуваної задачі.

Висновки

Зроблено огляд основних методів розв'язання задач комп'ютерного бачення, розглянуто їх переваги та недоліки. Теорія нечіткої логіки вивела розпізнавання на якісно новий рівень подарувавши новий методологічний та алгоритмічний фреймворк для роботи зі складними та недовизначеними системами.

Одним з найбільших вкладів теорії нечіткої логіки в розпізнавання образів є спрощення розробки алгоритмів на базі інтерпретованих моделей. Алгоритми на основі ймовірносних моделей погано інтерпретуються і тому їх неможливо доповнити експертними знаннями. Теорія нечітких множин має в своєму розпорядженні механізм, що дозволяє використовувати всі накопичені експертами знання в якості бази правил. Саме можливість використання експертних знань та інтерпретованість моделей є основними перевагами систем на базі нечіткої логіки. Використання множин другого типу може значно покращити отримані результати, проте суттєво збільшує обчислювальні затрати.

Проектування системи на базі нечіткої логіки може обернутись надзвичайно непростою задачею, оскільки з розширенням домену задачі буде збільшуватись і розмір бази правил та обчислювальні потреби. Через ресурсозатратність та складність проектування систем класифікації, теорію нечіткої логіки здебільшого використовують на попередніх етапах комп'ютерного бачення, таких як сегментація чи фільтрація.

Нейронні мережі є далеко не новим підходом, проте отримали значне поширення досить недавно з появою згорткових нейронних

мереж, що направлені саме на роботу з фото- та відеоданими. За допомогою рекурентних нейронних мереж реалізовано пошук архітектур нейромережевих систем. Різні моделі оптимізовано під різні обмеження, як то розмір моделі, обчислювальні затрати чи точність.

Для навчання нейромережевої системи зазвичай потрібні великі об'єми репрезентативних даних. Збір повноцінної бази зображень є нелегкою задачею. Варто зауважити, що найточніші архітектури є дуже вимогливими в плані пам'яті та обчислювальних потужностей, а швидким та невеликим моделям бракує точності.

Альтернативою попереднім двом підходам є використання особливостей самого зображення. Такий підхід не потребує великої бази зображень. Алгоритми на базі детекторів та дескрипторів використовуються в багатьох системах розпізнавання та до сплеску популярності нейронних мереж були найпоширенішим підходом до розв'язання задач класифікації.

Ефективним застосуванням детекторів та дескрипторів може бути автономна навігація безпілотних літальних апаратів, де потрібне співставлення ландшафтів та визначення поточного місцезнаходження. В такій задачі немає конкретних класів об'єктів, які потрібно розпізнати, замість розподілу на класи «дорога», «дерево» та ін., ще постає завдання співставлення, чи це та сама дорога, чи те саме дерево, координати якого відомі. Крім того, знімки можуть бути взяті з різної висоти.

Потрібно зазначити, що немає універсального детектора чи дескриптора, який переважає інші. Загалом цим методам бракує точності і їх застосування не є доцільним для всіх задач класифікації чи виявлення. Вибір конкретного апарату залежить від поставленої задачі. Для ефективного розв'язання тієї чи іншої задачі найкращим може бути поєднання декількох підходів.

Незважаючи на значний прогрес в галузі комп'ютерного бачення, все ще залишається багато відкритих питань. Одним з нагальних питань є застосування таких систем в умовах обмежених ресурсів — безпілотних літальних апаратах, мобільних пристроях, роботизованих та супутникових системах. Досягнення швидкодії реального часу при збереженні задовільної точності — це та задача, яка стоїть на даний момент перед дослідниками.

REFERENCES

1. Wang, C., Xu, A., Li, C., Zhao, X., 2016. "Interval type-2 fuzzy based neural network for high resolution remote sensing image segmentation," *ISPRS The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* Vols. XLI-B7, pp. 385—391.
2. Shi, J., Lei, Y., Zhou, Y., 2016. "A narrow band interval type-2 fuzzy approach for image segmentation," *Journal of Systems Architecture,* v. 64, pp. 86—99.
3. Murugeswari, P., Manimegalai, D., 2011. "Noise Reduction in Color image using Interval Type-2 Fuzzy Filter (IT2FF)," *International Journal of Engineering Science and Technology,* v. 3, 2, pp. 1334—1338.
4. Yuksel, M., Basturk, A., 2012. "Application of Type-2 Fuzzy Logic Filtering to Reduce Noise in Color Images," *IEEE Computer Intelligence Magazine,* v. 7, pp. 25—35.
5. Own, C. M., Tsai, H. H., Yu P.T., Lee, Y.J., 2006. "Adaptive type-2 fuzzy median filter design for removal of impulse noise," *Imaging Scientific Journal,* v. 54, pp. 3—18.
6. Melin, P. Mendoza O., Castillo, O., 2010. "An improved method for edge detection based on interval type-2 fuzzy logic," *Expert Systems with Applications,* v. 37, pp. 8527—8535.
7. Melin, P., Gonzalez, C., Castro, J., Mendoza, O., Castillo, O., 2014. "Edge-Detection Method for Image Processing Based on Generalized Type-2 Fuzzy Logic," *IEEE Transactions on Fuzzy Systems,* v. 22, pp. 1515—1525.
8. Gonzalez, C.I., Melin, P., Castillo, O., 2017. "Edge Detection Method Based on General Type-2 Fuzzy Logic Applied to Color Images," *Information (Switzerland),* vol. 8.
9. Lucas, L., Centeno, T., Delgado, M., 2008. "Land cover classification based on general type-2 fuzzy classifiers," *International Journal of Fuzzy Systems,* vol. 10, pp. 207—216.
10. Melin, P., 2018. "Genetic optimization of type-1, interval and intuitionistic fuzzy recognition systems," *Notes on Intuitionistic Fuzzy Sets,* v. 24, pp. 106—128.

11. CS231n Convolutional Neural Networks for Visual Recognition, [Online]. Available: http://cs231n.github.io/convolutional-networks. [Accessed 6 November 2018].
12. Krizhevsky, A., Sutskever I., Hinton, G.E., 2012. "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems (NIPS 2012)*.
13. Zeiler, M.D., Fergus, R. "Visualizing and Understanding Convolutional Networks," 2013. [Online]. Available: https://arxiv.org/abs/1311.2901v3. [Accessed 6 November 2018].
14. Simonyan, K., Zisserman, A. "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed 6 November 2018].
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. "Going deeper with convolutions," 2014. [Online]. Available: https://arxiv.org/abs/1409.4842. [Accessed 6 November 2018].
16. K. He, K., Zhang, X., Ren, S., Sun, J., 2016. "Deep residual learning for image recognition," *CVPR*.
17. Szegedy, C., Ioffe, S., Vanhoucke V., Alemi, A., 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. [Online]. Available: https://arxiv.org/abs/1602.07261. [Accessed 6 November 2018].
18. Veit, A., Wilber, M., Belongie, S. "Residual Networks Behave Like Ensembles of Relatively Shallow Networks," 2017. [Online]. Available: https://arxiv.org/abs/1605.06431v2. [Accessed 6 November 2018].
19. Abdi, M., Nahavandi, S. "Multi-Residual Networks: Improving the Speed and Accuracy of Residual Networks," 2017. [Online]. Available: https://arxiv.org/abs/1609.05672v4. [Accessed 6 November 2018].
20. Zagoruyko, S., Komodakis, N., 2017. Wide Residual Networks. [Online]. Available: https://arxiv.org/abs/1605.07146v4. [Accessed 6 November 2018].
21. Larsson, G., Maire, M., Shakhnarovi, G. "FractalNet: Ultra-Deep Neural Networks without Residuals," 2017. [Online]. Available: https://arxiv.org/abs/1605.07648v4. [Accessed 6 November 2018].
22. Iandola, F.N., Han, S., Moskewicz, M.W., Dally, W.J., Keutzer, K., 2016. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". [Online]. Available: https://arxiv.org/abs/1602.07360. [Accessed 6 November 2018].
23. Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 580—587.
24. Girshick, R., 2015. Fast R-CNN. [Online]. Available: https://arxiv.org/abs/1504.08083v2. [Accessed 6 November 2018].
25. Ren, S., He, K., Girshick, R., Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015. [Online]. Available: https://arxiv.org/abs/1506.01497. [Accessed 6 November 2018].
26. Dai, J., Li, Y., He, K., Sun, J., 2016. R-FCN: Object Detection via Region-based Fully Convolutional Networks, [Online]. Available: https://arxiv.org/abs/1605.06409. [Accessed 6 November 2018].
27. Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2015. You only look once: Unified, real-time object detection, [Online]. Available: https://arxiv.org/abs/1506.02640. [Accessed 6 November 2018].
28. Liu, W., Anguelov, D., Erhan, D., Szegedy, C. Reed, S., Fu, C.-Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector, [Online]. Available: https://arxiv.org/abs/1512.02325v5. [Accessed 6 November 2018].
29. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K., 2017. Speed/accuracy trade-offs for modern convolutional object detectors. [Online]. Available: https://arxiv.org/abs/1611.10012v3. [Accessed 6 November 2018].
30. Kyyko, V.M., Matsello, V.V., 2015. "The Fingerprints Recognition Based on Corresponding Points Searching". *Upravlausie sistemy i masiny*, 3, pp. 36—41. (In Russian).
31. Fischler, M.A., Bolles, R.C., 1981. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM*.
32. Moravec, H., 1980. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover". Tech Report CMU-RI-TR-3," Carnegie-Mellon University, Robotics Institute.
33. Canny, J., 1986. "A Computational Approach To Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1986.
34. Harris, C., Stephens, M., 1988. "A combined corner and edge detector," *Proceedings of the 4th Alvey Vision Conference*.
35. Rosten, E., Drummond, T., 2006. Machine Learning for High-speed Corner Detection.
36. Rosten, E., 2008. Faster and better: a machine learning approach to corner detection.
37. Mair, E., Hager, G.D., Burschka, D., Suppa, M., Hirzinger, G. "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test," *European Conference on Computer Vision (ECCV'10)*, September 2010.
38. Lowe, David, G., 1999. "Object recognition from local scale-invariant features," *Proceedings of the International Conference on Computer Vision*.

39. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., 2008. "Speeded Up Robust Feature," *ETH Zurich, Katholieke Universiteit Leuven*.
40. Calonder, M., Lepetit, V., Strecha, C., Fua, P., 2010. "BRIEF: Binary Robust Independent Elementary Features," *11th European Conference on Computer Vision (ECCV)*.
41. Leutenegger, S., Chli, M. Siegwart, R.Y., 2011. "BRISK: Binary Robust invariant scalable keypoints," *2011 International Conference on Computer Vision*, Barcelona, Spain.
42. Rublee, E., Rabaud, V., Konolige, K.M, Bradski, G., 2011. "ORB: An efficient alternative to SIFT or SURF," *2011 International Conference on Computer Vision*, Barcelona, Spain.
43. Alcantarilla, P.F., Bartoli, A., Davison, A.J., 2012. "KAZE Features," in *European Conference on Computer Vision 2012 (ECCV 2012)*.
44. Alcantarilla, P.F., Nuevo, J., Bartoli, A., 2013. Fast explicit diffusion for accelerated features in nonlinear scale spaces, BMVC.
45. Roos, D.R., Shiguemori, E.H., Lorena, A.C., 2016. "Comparing ORB and AKAZE for visual odometry of unmanned aerial vehicles," *4th Conference of Computational Interdisciplinary Sciences, 2016.*
46. Byrne, J., Laefer, D.F., O'Keffe, E., 2017. "Maximizing feature detection in aerial unmanned aerial vehicle datasets," *Journal of Applied Remote Sensing,* 11(2).
47. Isık, S. Özkan, K., 2014. "A Comparative Evaluation of Well-known Feature Detectors and Descriptors.," *International Journal of Applied Mathematics, Electronics and Computers,* v. 3.
48. Tareen, S.A.K., Saleem, Z., 2018. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK.," *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*.

Received 22.11.2018

Tymchyshyn Roman, PhD student,
International Research and Training Center for Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03187, Ukraine,
romantymchyshyn.rt@gmail.com

Volkov Olexander, head of department,
International Research and Training Center for Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03187, Ukraine,
alexvolk@ukr.net

Gospodarchuk Oleksiy, senior research fellow,
International Research and Training Center for Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03187, Ukraine,
dep185@irtc.org.ua

Bogachuk Yuriy, leading research fellow,
International Research and Training Center for Information Technologies and Systems of the NAS and MES of Ukraine, Glushkov ave., 40, Kyiv, 03187, Ukraine,
dep185@irtc.org.ua

MODERN APPROACHES TO COMPUTER VISION

Introduction. Computer vision includes a wide variety of problems: image segmentation, processing, classification, scene reconstruction, pose estimation, object detection, trajectory tracking and others. These problems are cornerstones of artificial intelligence.

The field has been rapidly evolving in recent years, partly due to the fact that such giants of IT industry as Google and Microsoft have joined the research. AI systems are in high demand nowadays. Technological advances have enabled many applications of computer vision in dozens of industries. Among them are such well known applications as smart stores, biometric authentication, automation of agricultural processes using drones, video surveillance, improving the quality of photo and video data, autonomous delivery of parcels by unmanned aerial vehicles. The scope will be expanding since the need for artificial intelligence systems increases over time and vision is one of the most informative sensors that can be used in such systems.

Purpose. The number of developments in the field of computer vision increases exponentially and staying up to date is not an easy task. There is a wide variety of existing approaches and choosing the right one can be difficult. The goal of this

paper is to present a structured overview of modern techniques in the field of computer vision with their advantages and disadvantages, and identification of unresolved problems. Accuracy is not the only quality measure considered, we also take speed and memory into account, which is critical for embedded systems (unmanned aerial vehicles, mobile devices, robotic and satellite systems).

Methods. Fuzzy logic, convolutional neural networks, feature detectors and descriptors.

Results. Fuzzy logic theory has led recognition to a completely new level by presenting a new methodological and algorithmic framework for working with complex and uncertain systems. Introduction of type-2 fuzzy sets has significantly improved accuracy and robustness. Their main advantages are the use of expert's knowledge and interpretability of fuzzy logic models. Now fuzzy logic is mainly used as a complement for other systems with the aim to improve decision making process by handling the uncertainty. Researchers often employ this technique for solving image segmentation and filtering problems.

Convolutional neural networks (CNN) make the explicit assumption that the inputs are images. This assumption allows to encode certain properties into the architecture and lead to striking results. CNN architectures even managed to beat human in a classification task in some cases (e.g. on ImageNet visual database). Presented here are the architectures with state-of-the-art results in image classification and object detection tasks.

Feature detectors and descriptors were the most commonly used tool in image processing for years. They remain a great alternative to resource intensive neural networks. Methods based on feature detectors and descriptors don't require large databases for learning. A good fit for these types of methods is autonomous navigation of unmanned aerial vehicles where images matching is needed for the coordinate identification.

Conclusion. While the great progress has been made in recent years there is still a number of unsolved problems. Existing algorithms lack generality. Performance improvement usually leads to accuracy degradation. There are no high-quality accurate algorithms that can solve object detection problems in real-time. Use of accurate computer vision algorithms requires significant amounts of memory and computing resources that may not be available on embedded systems. Training time of deep convolutional neural networks is still large and can reach weeks even on the most performant computers. There is no clear way to deal with low quality images.

Keywords: Computer vision, image classification, object detection, image segmentation, image filtering, edge detection, fuzzy logic, neural networks, feature detectors, feature descriptors.

Р.М. Тимчишин, аспирант, отдел интеллектуального управления,
Международный научно-учебный центр информационных технологий и систем
НАН Украины и МОН Украины, просп. Академика Глушкова, 40, Киев 03187, Украина,
romantymchyshyn.rt@gmail.com

А.Е. Волков, заведующий отделом интеллектуального управления,
Международный научно-учебный центр информационных технологий и систем
НАН Украины и МОН Украины, просп. Академика Глушкова, 40, Киев 03187, Украина,
alexvolk@ukr.net

А.Ю. Господарчук, старший научный сотрудник, отдел интеллектуального управления,
Международный научно-учебный центр информационных технологий и систем
НАН Украины и МОН Украины, просп. Академика Глушкова, 40, Киев 03187, Украина,
dep185@irtc.org.ua

Ю.П. Богачук, ведущий научный сотрудник, отдел интеллектуального управления,
Международный научно-учебный центр информационных технологий и систем
НАН Украины и МОН Украины, просп. Академика Глушкова, 40, Киев 03187, Украина,
E-mail: dep185@irtc.org.ua

СОВРЕМЕННЫЕ ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Введение. Компьютерное зрение включает в себя широкий спектр задач: сегментацию, обработку и классификацию изображений, восстановление сцен, обнаружение, оценку положения и восстановление траекторий объектов, и многие другие.

Эта нгаправление стремительно развивается в последние годы, частично и потому, что к исследованию присоединились такие гиганты ИТ-индустрии, как *Google* и *Microsoft*. В настоящее время системы искусственного

интеллекта пользуются большим спросом. Технологические достижения сделали возможным применение компьютерного зрения в десятках новых отраслей, таких как умные магазины, биометрическая аутентификация, автоматизация сельскохозяйственных процессов с использованием беспилотных летательных аппаратов (БпЛА), видеонаблюдение, улучшение качества фото- и видеоданных, автономная доставка посылок. Область применения будет расширяться, поскольку потребность в системах искусственного интеллекта только возрастает.

Цель. Количество наработок в области компьютерного зрения увеличивается экспоненциально и выбор подходящего инструмента — непростая задача. Цель данной статьи — представить структурированный обзор современных технологий компьютерного зрения с их преимуществами и недостатками, а также идентифицировать нерешенные проблемы.

Методы. Нечеткая логика, сверточные нейронные сети, детекторы и дескрипторы ключевых точек.

Результаты. Теория нечеткой логики вывела распознавание на качественно новый уровень представив новый фреймворк для работы со сложными и неопределенными системами. Введение нечетких множеств второго типа значительно улучшило точность и устойчивость алгоритмов. Основными преимуществами систем на базе нечеткой логики являются использование знаний эксперта и интерпретируемость финальной модели. Сейчас этот метод часто используется для решения задач сегментации и фильтрации изображений.

Сверточные нейронные сети (*CNN*) делают явное предположение о том, что на вход системе подается изображение. Это предположение позволило внедрить определенные свойства в архитектуру и привело к поразительным результатам. Архитектура *CNN* даже в некоторых случаях превзошла человека в задаче классификации (например, на базе данных *ImageNet*). В работе представлены архитектуры с лучшими результатами в задачах классификации изображений и обнаружения объектов.

Детекторы и дескрипторы ключевых точек — наиболее часто используемый инструмент обработки изображений в течение многих лет. Они остаются отличной альтернативой ресурсоемким нейронным сетям. Методы, что базируются на детекторах и дескрипторах, не требуют больших баз данных для обучения. Хорошим применением этих типов методов может быть автономная навигация беспилотных летательных аппаратов, где для идентификации координат требуется сопоставление изображений.

Выводы. Несмотря на то, что в последние годы удалось добиться значительного прогресса, все еще остается множество нерешенных проблем. Повышение производительности обычно приводит к ухудшению точности. Нет достаточно точных алгоритмов, которые могли бы решать проблемы обнаружения объектов в режиме реального времени. Использование точных алгоритмов компьютерного зрения требует значительных объемов памяти и вычислительных ресурсов, которые недоступны во встраиваемых системах. Время обучения глубоких сверточных нейронных сетей по-прежнему велико и может достигать недель даже на самых производительных компьютерах.

Ключевые слова: Компьютерное зрение, классификация изображений, обнаружение объектов, сегментация изображений, фильтрация изображений, выделение границ, нечеткая логика, нейронные сети, детекторы, дескрипторы.