Т. 1, № 1, 2019

https://doi.org/10.23939/csn2019.01.024

УДК 004.415.2

МЕТОДИ ПОШУКУ ТА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОЗОБРАЖЕННЯХ НА МОБІЛЬНІЙ ПЛАТФОРМІ IOS В РЕАЛЬНОМУ ЧАСІ

Д. О. Кушнір, Я. С. Парамуд

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

Досліджено особливості найпоширеніших методів і систем пошуку та розпізнавання об'єктів у відеозображеннях. За результатами дослідження показано доцільність побудови засобів пошуку та розпізнавання для платформи iOS у реальному часі. Запропоновано метод функціональної адаптації алгоритму пошуку та розпізнавання об'єктів до особливостей відеозображень, який полягає в опрацюванні відеозображення згладжуючим та мінімізаційним фільтрами, що забезпечує зменшення часу пошуку та розпізнавання об'єктів. Розроблено базову структурну схему таких засобів та алгоритм функціонування. Розроблено алгоритмічно-програмні засоби для розв'язання завдання на знаходження та оперативне розпізнавання об'єктів у режимі реального часу мовою Swift під мобільну платформу iOS. Використано особливості згорткової нейронної мережі з архітектурою YOLOv3 та фреймворку для роботи з нейронними мережами під мобільні додатки CoreML. Запропоновано метод поліпшення роботи такої нейронної мережі, який оснований на квантизації вагових коефіцієнтів нейромережі та забезпечує мінімізацію розміру моделі та часу пошуку її об'єктів. Досліджено значення частоти оброблення кадрів зображень із використанням запропонованої моделі YOLOv3-KD та моделей нейронних мереж типу YOLOv3-tiny та YOLOv3-416. Доведено можливість функціонування запропонованих засобів у режимі реального часу.

Ключові слова: час пошуку об'єктів, час розпізнавання об'єктів, відеозображення, мобільна платформа, згорткова нейронна мережа, реальний масштаб часу.

Вступ

Широке використання сучасних мобільних засобів, збільшення їх функціональних можливостей забезпечує розв'язання різних класів завдань. Одним із них є пошук і розпізнавання певного класу об'єктів на зображенні чи відеопотоці для подальшого його опрацювання. Сьогодні розроблено низку засобів реалізації цього завдання [1, 2]. Проте залишається проблема забезпечення якості оброблення залежно від таких факторів, як: швидкість опрацювання великих обсягів даних, пропускна здатність мережі, якщо машина навчання знаходиться віддалено, вірогідність знаходження об'єктів, складність тренування початкових моделей пошуку і класифікації об'єктів.

Водночас на щораз більшому ринку мобільного програмного забезпечення переважають кросплатформенні додатки, тобто розроблені під декілька платформ. Незважаючи на свою поширеність, вони мають певні недоліки, основними з яких є важка переносимість коду, обмежений програмний функціонал, обмежена можливість функціонування в реальному часі та погана відмовостійкість.

Беручи до уваги наведене вище, можна стверджувати, що дослідження стосовно розроблення ефективних методів пошуку та розпізнавання об'єктів у відеозображеннях своєчасні та актуальні.

Огляд літературних джерел

Сьогодні розроблено низку методів і систем пошуку та розпізнавання об'єктів на відеозображеннях, в яких широко застосовані згорткові нейронні мережі (ЗНМ). Використано декілька методів машинного навчання для вирішення проблеми пошуку та розпізнавання. Серед них можна виокремити:

- *Метод бінарної класифікації ознак зображення* [1] – завдання класифікації, що є предметом розгляду в машинному навчанні. Це завдання навчання з учителем – метод, в якому категорії відомі, полягає у визначенні цих категорій для нових спостережень. Коли таких категорій всього дві, то це статистична бінарна класифікація;
- *Метод опорних векторів ознак (ОВО) зображення* [2] – це метод аналізу даних для класифікації та регресійного аналізу за допомогою моделей з керованим навчанням із пов'язаними алгоритмами навчання, які називаються опорно-векторними машинами (ОВМ). Для заданого набору тренувальних зразків, кожен із яких виокремлено як належний до однієї чи іншої з двох категорій, алгоритм тренування ОВМ будує модель, яка зараховує нові зразки до однієї чи іншої категорії, роблячи це неймовірнісним бінарним лінійним класифікатором;
- *Метод штучної нейронної мережі (ШНМ) ознак зображення* [3] – ґрунтується на сукупності з'єднаних вузлів, що називають штучними. Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його й потім сигналізувати штучним нейронам, приєднаним до нього. Використання штучних нейронних мереж для класифікації зображень – один із доволі ефективних методів. Проте така класифікація ускладнюється, насамперед, великим розміром вектора вхідних зображень нейронної мережі, кількістю нейронів у проміжних шарах, і, як наслідок, великими обчислювальними витратами на навчання і виконання мережі;
- *Метод згорткової нейронної мережі (ЗНМ) вхідного зображення* [4] – це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоби вимагати використання мінімального обсягу попереднього оброблення. Цей метод дуже часто використовують із високою ефективністю під час оброблення фото- та відеозображення, особливо в розпізнаванні таких категорій, як порода собак чи птахів, з чим у пересічної людини можуть виникати проблеми.

Порівняльний аналіз переваг та недоліків розглянутих методів дає змогу встановити, що для вирішення завдання з пошуку і розпізнавання об'єктів у відеозображеннях доцільно використовувати ЗНМ. Сьогодні існує багато ЗНМ, які вирішують ці завдання. До найпоширеніших можна зарахувати такі:

- *R-CNN (Regions with CNN)* [5]. Метод пошуку об'єктів, який працює як звичайний класифікатор зображень. На вхід мережі подають різні регіони зображення і для них роблять передбачення. У результаті процедура дуже повільна, оскільки проганяє одне зображення кілька тисяч разів.
- *Fast R-CNN* [6]. Поліпшення і швидша версія R-CNN, що працює за схожим принципом. Спочатку все зображення подають на вхід згорткової нейронної мережі, потім з отриманого внутрішнього представлення генерують регіони. Але, як і в попередньому випадку, мережа доволі повільна для задач реального часу.
- *YOLO* [7]. Використовує зовсім інший принцип роботи порівняно з попередніми системами. Тоді як більшість систем використовують згорткову нейронну мережу декілька разів із різними регіонами зображення, YOLO використовує її лише один раз до всього зображення. Мережа ділить зображення на сітку і передбачає розташування потрібного об'єкта для кожної ділянки. YOLO істотно швидше за R-CNN, що є критично важливо для оброблення інформації в режимі реального часу на мобільному пристрої. Має багато реалізацій у вигляді бібліотек, таких як Keras [8].

- *SSD* [9]. За принципом схожа на YOLO, але як мережа для вилучення ознак використовує VGG16 [10]. Теж доволі швидка і придатна для роботи в реальному часі, проте розмір вихідної моделі істотно більший, ніж у YOLO.

Серед цілісних продуктів, які використовують ЗНМ, можна виокремити:

– *EyeSpy* [11] – програма розпізнавання об'єктів на фотографії та пошук релевантної інформації в мережі інтернет. Недолік – відсутня підтримка режиму в реальному часі.

– *Google Translate* [12] – програма для розпізнавання та перекладу тексту в режимі реального часу. Кожну мову потрібно завантажити окремим файлом.

Недоліками більшості розглянутих систем, насамперед *EyeSpy*, є потреба використовувати інтернет з'єднання для отримання та відправлення інформації на опрацювання до машини навчання. За порівняно невеликих розмірів вихідної програми вони залежать від латентності з'єднання з мережею.

Постановка завдання

За результатами дослідження наявних засобів розробити ефективний метод та засоби пошуку та розпізнавання об'єктів у відеозображеннях на мобільній платформі iOS в реальному часі.

Основні результати досліджень

Запропоновано метод функціональної адаптації алгоритму пошуку та розпізнавання об'єктів до особливостей відеозображень, який полягає в опрацюванні відеозображення згладжувальним та мінімізаційним фільтрами, що забезпечує зменшення часу пошуку та розпізнавання об'єктів.

Загалом розв'язання поставленого завдання потребує таких кроків:

- створення запропонованої моделі згорткової нейронної мережі з новими класами об'єктів на основі архітектури YOLOv3;
- конвертування вагових коефіцієнтів у формат, зрозумілий мобільному фреймворку для роботи машини навчання CoreML;
- створення додатку на iOS та інтеграція моделі CoreML;
- накладання мінімізаційного та згладжувального фільтрів;
- забезпечення роботи системи в режимі реального часу з достатньою частотою кадрів;
- забезпечення можливості динамічної зміни кількості розпізнавальних об'єктів, використовуючи різні моделі нейронної мережі.

Запропоновану схему, що реалізовує наведені вище кроки розроблення поставленого завдання, наведено на рис. 1.

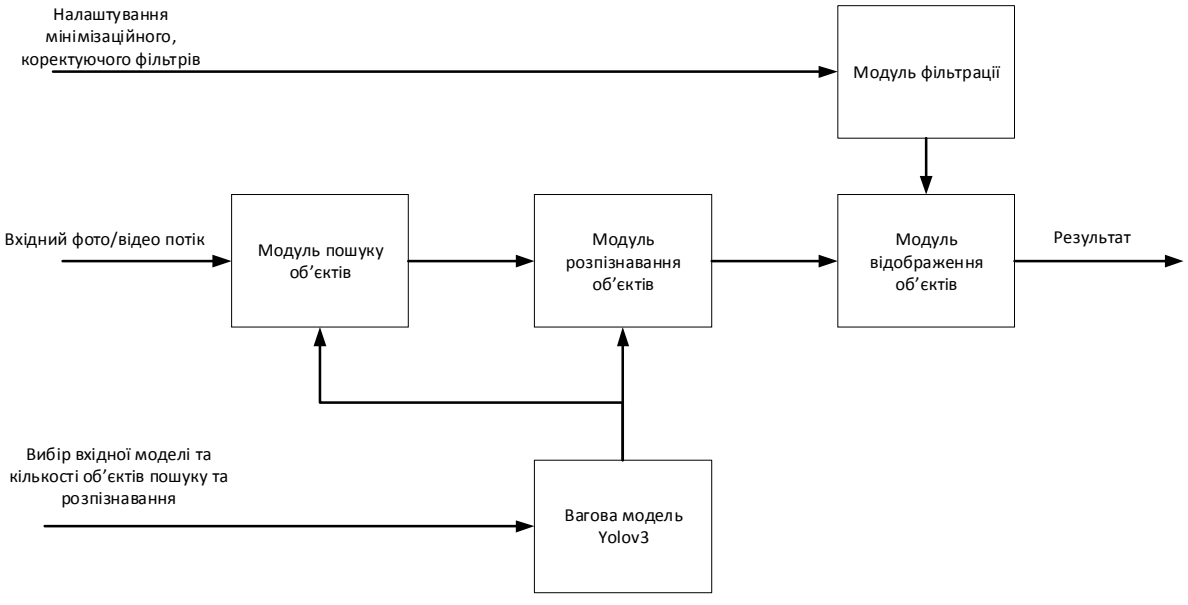

Рис. 1. Структурна схема системи пошуку та розпізнавання об'єктів

Реалізацію запропонованого методу та системи пошуку і розпізнавання об'єктів у відеозображеннях на мобільній платформі iOS в реальному часі можна здійснити із використанням таких шести етапів та відповідно до схеми алгоритму, наведеної на рис. 2. Передбачено початкове налаштування мінімізаційного та коректувальних фільтрів, які просто впливають на якість вихідного відображення обробленної моделі. Вхідне зображення з фото/відеопотоку проходить через модулі пошуку та розпізнавання, та, використовуючи згенеровану вагову модель, передає результати оброблення до модуля розпізнавання. Після цього відбувається функціональна адаптація алгоритму з використанням згаданих вище фільтрів. Результат виводиться на екран.

1. Побудова граф схеми алгоритму роботи системи

На рис. 2 продемонстровано схему алгоритму роботи системи.

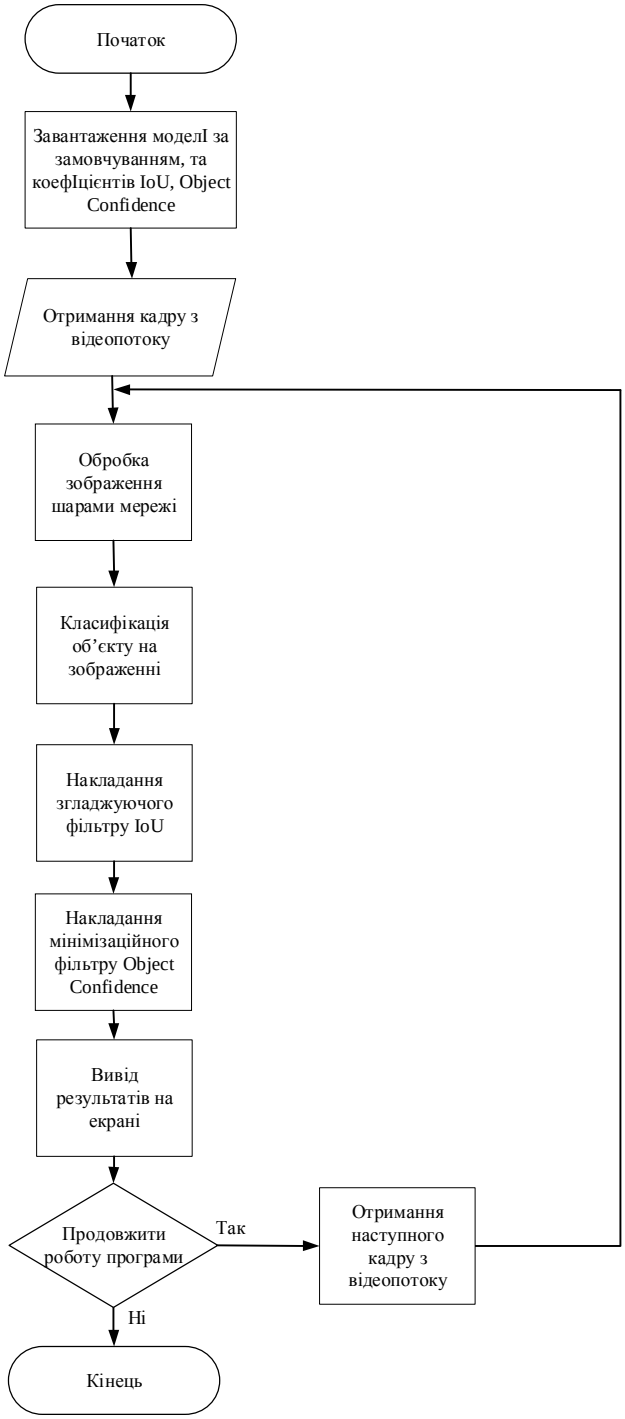

Рис. 2. Схема алгоритму роботи системи

За результатами розпізнавання отримуємо набір передбачень, який формується послідовно на екрані пристрою для кожного кадру з відеопотоку. Коректуючі фільтри мінімізують та згладжують результати передбачень для конкретної моделі.

2. Конвертування моделей YOLOv3 *у формат CoreML.*

Оригінальну архітектуру YOLOv3 реалізовано за допомогою фреймворку Darknet. На мобільних пристроях iOS, починаючи з версії 11.0, доступний фреймворк CoreML, який дає змогу інтегрувати моделі машинного навчання у мобільний пристрій.

Проте виникає проблема в тому, що CoreML розуміє тільки певний формат – .coreml. Водночас у фреймворку Darknet наразі не реалізовано пряму конвертацію між цими двома моделями. Отже, для того щоб перетворити збережену і навчену модель із Darknet у CoreML, необхідно спершу конвертувати Darknet модель у формат Keras. Далі конвертовану модель буде перетворено в формат зрозумілий CoreML.

Для першої конвертації з yolov3 в keras необхідно виконати команду:

python convert.py yolov3.cfg yolov3.weights yolo.h5,

де yolov3.cfg і yolov3.weights – вхідна модель Darknet. У результаті згенерується файл із розширенням .h5 формату Keras.

Для другої конвертації необхідно запустити наступний скрипт, використовуючи бібліотеку coremltools для роботи з CoreML:

```
import coremltools
coreml_model = coremltools.converters.keras.convert(
  'yolo.h5',
  input_names='image',
  image_input_names='image',
  input_name_shape_dict={'image': [None, 416, 416, 3]}, #розмір вхідного зображення для мережі
  image_scale=1/255.)
coreml_model.input_description['image'] = 'Input image'
coreml_model.save('yolo.mlmodel')
```

Після цього згенеруються моделі формату .mlmodel, які можна використовувати в роботі аплікації. Процедуру конвертації необхідно виконати для усіх використаних моделей нейронних мереж: YOLOv3-tiny, YOLOv3-416 та YOLOv3-KD.

3. Аналіз роботи моделей типу YOLOv3

YOLOv3 – принцип роботи пояснюється структурною схемою, наведеною на рис. 3. Це вдосконалена версія архітектури YOLO. Вона складається із 106 згорткових шарів. Моделі YOLO навчені на основі глобальної бази даних зображень. Основна особливість YOLOv3 полягає в тому, що на виході є три шари, кожен із яких розрахований на виявлення об’єктів різного розміру.

YOLOv3-tiny – спрощена версія архітектури YOLOv3, що складається з двох вихідних шарів. Вона гірше передбачає дрібні об’єкти і призначена для невеликих об’ємів даних. Але, через особливості побудови, ваги мережі займають невеликий обсяг пам’яті (приблизно 35 МБ) і вона зменшує час опрацювання кадрів. Тому така архітектура доречніша для використання у мобільному пристрої.

YOLOv3-KD – запропонована за результатами досліджень модель нейронної мережі. Аналогічно до Yolov3 має 3 вихідні шари, проте орієнтована на меншу кількість об’єктів. Має перевагу перед попередніми версіями у швидкості та розмірі пам’яті.

4. Тренування запропонованої моделі YOLOv3-KD

Структурну схему алгоритму тренування моделі показано на рис. 4.

Кожна ітерація повторюється регресивною спробою опрацювати вхідне зображення із використанням еталонів. Що вдаліша спроба, то ліпший результуючий коефіцієнт. Якість тренування зображення залежить від точності маркування вхідних зображень та кількості зображень.

Процес та результати тренування моделі наведено на рис. 5.

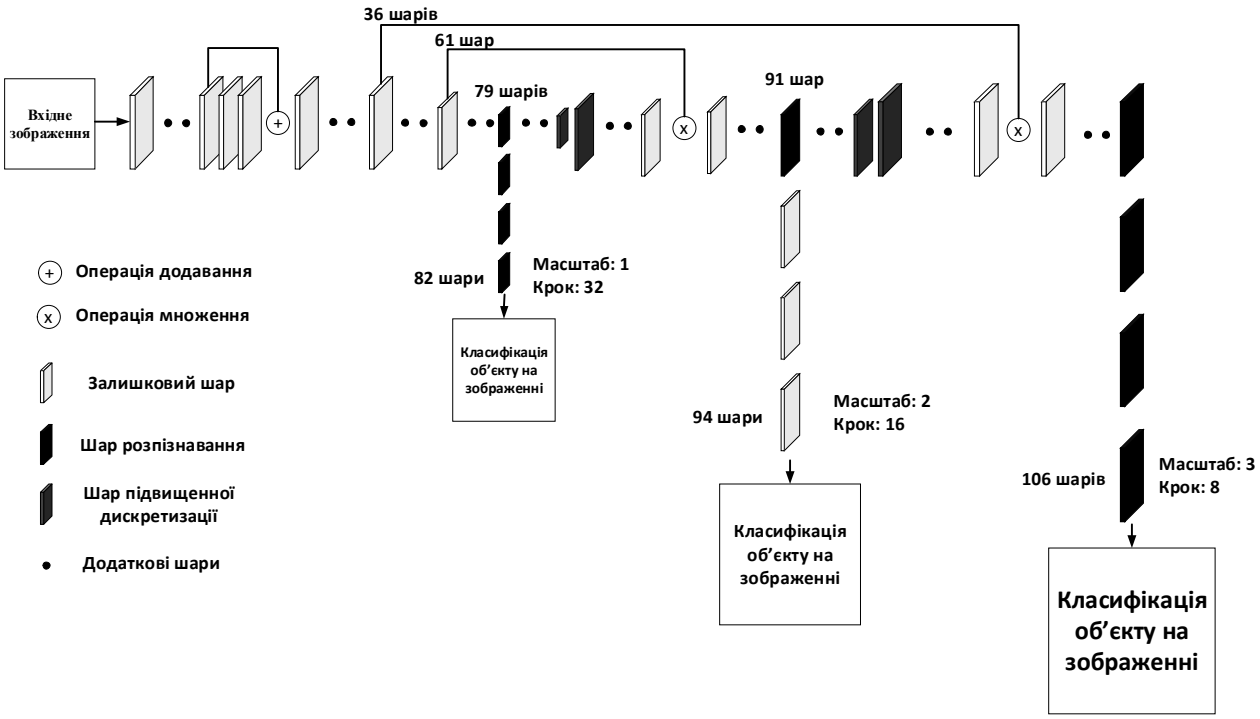

Рис. 3. Структурна схема архітектури YOLOv3

Рис. 4. Структурна схема алгоритму тренування моделі

Крива вказує кількість помилок, що виникають під час навчання, та демонструє, чи мережа вчиться (графік прямує вниз) чи деградує (графік прямує угору). По осі ***y*** ми маємо значення Loss (Kv) – коефіцієнт втрат, по осі ***x*** – кількість пройдених ітерацій (Ni). Навчання зупинено на 4000 ітерації та вибрано як результуючі вагові коефіцієнти 4000 ітерацію, оскільки цього виявилось цілком достатньо для двох об'єктів та коефіцієнт Kv показував найліпші результати. Над вихідною моделлю проведено процедуру квантизації вагових коефіцієнтів, що призвело до зменшення витрат пам'яті зі 120 МБ до майже 20 МБ.

5. Використання фільтрів для поліпшення результатів вихідної моделі

Після завантаження моделей у систему розроблення Xcode можна спостерігати отримані вихідні шари кожної з них. Результати запропонованої моделі продемонстровано на рис. 6.

Відповідно до рисунку вище, до розроблюваної моделі YOLOv3-KD включено три вихідні шари, кожному з яких передбачено регіони для різних об'єктів у вигляді двовимірного масиву. Для відомих моделей Yolov3-416 та Yolov3-Tiny запропоновано три та два вихідних шарів відповідно. Розміри комірок масиву мають такі значення: 8, 16 і 32. Припустимо на вході у нас є зображення

розміром 416×416 пікселів. Тоді вихідні матриці (сітки) матимуть розмір: 52×52, 26×26 та 13×13 (416/8 = 52, 416/16 = 26 і 416/32 = 13).

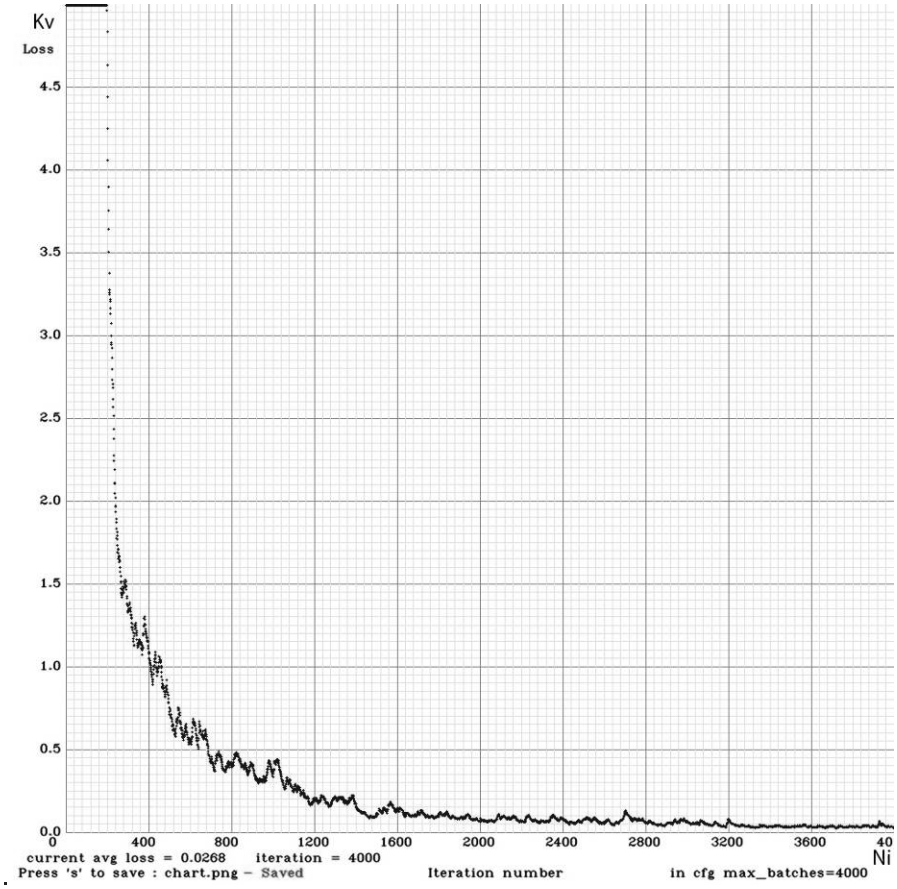

Рис. 5. Графік процесу тренування запропонованої моделі нейронної мережі

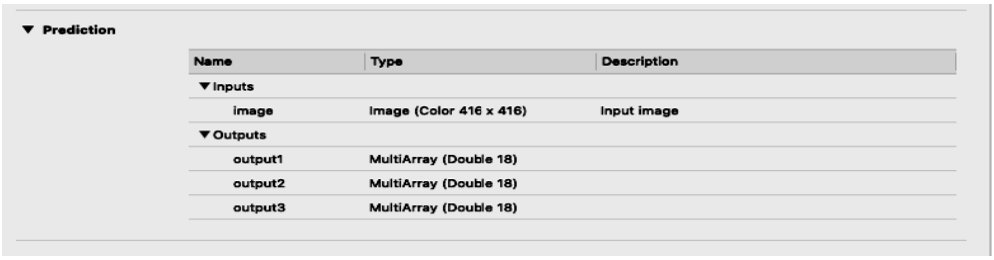

Рис. 6. Вхідні та вихідні шари моделі YOLOv3-KD

Після запуску завантаженої моделі CoreML для розроблюваної моделі нейронної мережі на виході отримуємо такі результати: [1, 1, 18, 25, 25]. Де 18 – це вектор, який отримують за формулою:

$$A = B * ((t_x + t_{y+} t_{w+} t_{h+} p_o) + (p_1 + p_2 + \ldots + p_n)), \quad (1)$$

де A – результуючий вектор; B – вихідні шари мережі (регіони); t_x, t_y, t_w, t_h – координати x, y, ширина та висота передбачуваного регіону відповідно; p_o – коефіцієнт передбачення появи регіону (мінімізаційний фільтр Object confidence); $p_1..p_n$ – вхідні класи, n – кількість вхідних класів.

Тепер потрібно обробити цей вектор. Для кожного передбачуваного регіону потрібно отримати розподілення вірогідності щодо заданого діапазону класів. Зробити це можна за допомогою функцій softmax. Приклад реалізації на Swift:

```
private func softmax(_ x: inout [Float]) {
  let len = vDSP_Length(x.count)
```

```
    var count = Int32(x.count)
    vvexpf(&x, x, &count)
    var sum: Float = 0
    vDSP_sve(x, 1, &sum, len)
    vDSP_vsdiv(x, 1, &sum, &x, 1, len)
  }
```

Для отримання координат і розмірів регіонів потрібно скористатися формулами:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \quad , \tag{2}$$

де b_x, b_y, b_w, b_h – передбачені x, y координати, ширина і висота відповідно; t_x, t_y, t_w, t_h – виходи нейронної мережі; $\sigma(x)$ – функція сигмоїд; *cx* та *cy* – верхні крайні ліві початкові координати сітки, а *pw* та *ph* – значення якорів для трьох регіонів. Ці значення визначають під час тренування і задають завчасно, наприклад значення ph та pw:

```
let anchors1: [Float] = [116,90, 156,198, 373,326] // якоря для першого вихідного шару,
let anchors2: [Float] = [30,61, 62,45, 59,119] // якоря для другого вихідного шару,
let anchors3: [Float] = [10,13, 16,30, 33,23] // якоря для третього вихідного шару.
```

Структурну схему формування регіону подано на рис. 7.

Отримавши координати і розміри регіонів та відповідних ймовірностей для всіх знайдених об'єктів на зображенні, можна починати відмальовувати їх поверх картинки.

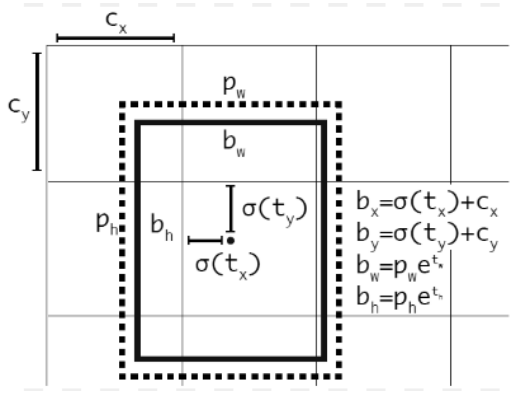

Рис. 7. Структурна схема розрахунку положення регіону

Проте існує ймовірність виникнення проблеми, коли для одного об'єкта передбачено кілька регіонів з доволі високими ймовірностями. Для вирішення цієї проблеми доцільно використати алгоритм *Non-Max Suppression*:

- Шукаємо регіон із найбільшою ймовірністю приналежності до об'єкта.
- Переглядаємо всі регіони, які теж належать до цього об'єкта.
- Видаляємо їх, якщо *Intersection over Union* (IoU) з першим регіоном більший ніж заданий поріг.

Згладжувальний фільтр IoU вираховують за формулою:

$$\text{IoU} = \text{Sp} / \text{So}\,, \tag{3}$$

де Sp – площа перетину, So – площа об'єднання.

Приклад реалізації мовою Swift:

```
static func IOU(a: CGRect, b: CGRect) -> Float {
  let areaA = a.width * a.height
  if areaA <= 0 { return 0 }
  let areaB = b.width * b.height
  if areaB <= 0 { return 0 }
  let intersection = a.intersection(b)
  let intersectionArea = intersection.width * intersection.height
  return Float(intersectionArea / (areaA + areaB - intersectionArea))
}
```

6. Виведення результатів із використанням мобільного додатку

Додаток матиме дві можливості виведення інформації: через камеру телефону в режимі реального часу або ж через фотопотік (з камери чи з галереї). Також буде можливість налаштування типу моделі, порогу ***Intersection over Union*** *(мінімізаційний фільтр),* порогу ***Object Confidence*** *(згладжуючий фільтр)* та можливості додаткового згладження вихідних регіонів на етапі кінцевого оброблення. Результати роботи наведено на рис. 8.

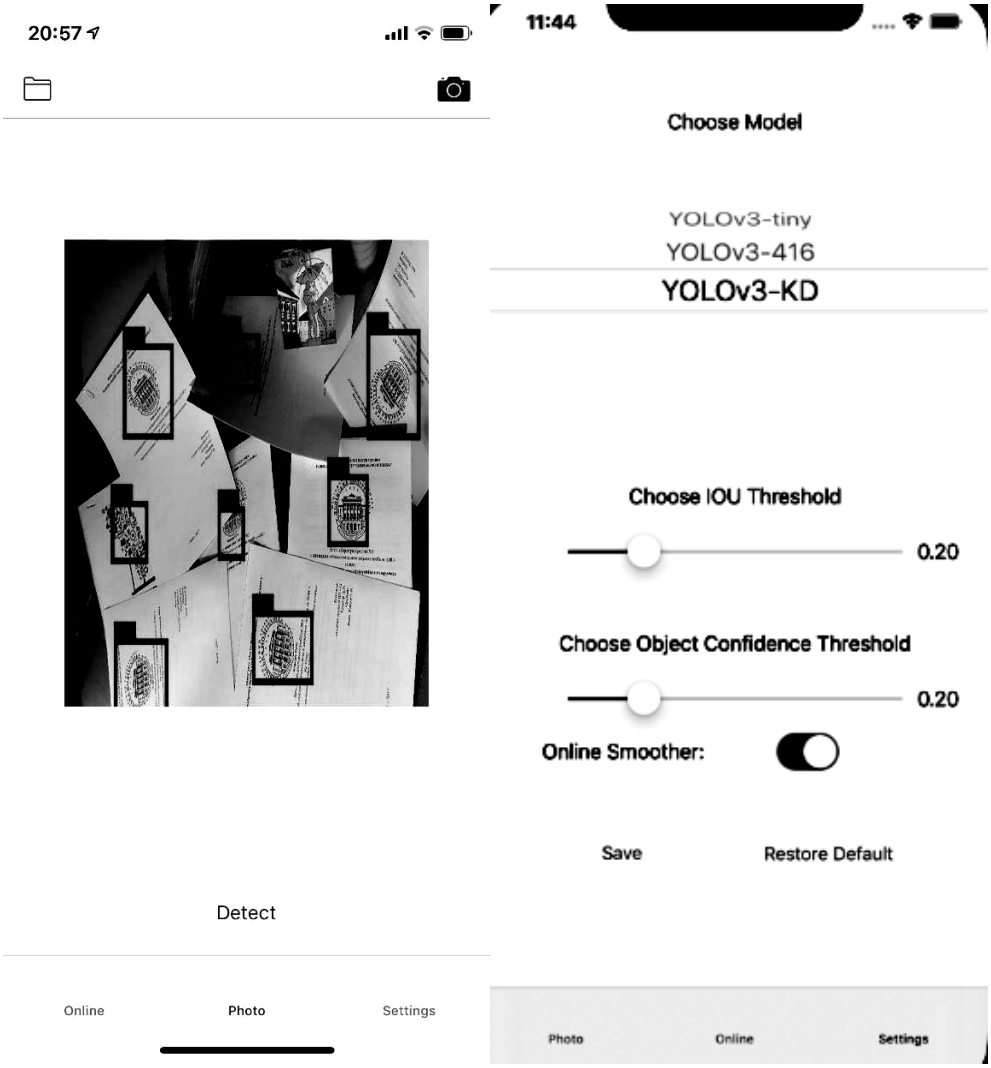

Рис. 8. Результати розпізнавання емблеми Львівської політехніки (зліва) та вибір моделі нейронної мережі (справа)

У процесі розпізнавання використано пороги object confidence та IoU, тобто накладання алгоритму Non-Max Suppression. Online smoother – це додаткова опція для корекції накладання регіонів на об’єкт зображення. Також варто зауважити, що емблему здебільшого розпізнано правильно.

Розроблено модель пошуку та розпізнавання об’єктів порівняно із близькими аналогами. Досліджено частоту кадрів, час розпізнавання одного регіону та необхідну ємність пам’яті відповідно для моделі нейронної мережі та обраного мобільного пристрою. Результати наведено в таблиці.

Як видно з результатів, наведених у таблиці, запропонована модель нейронної мережі має переваги у швидкості та розмірі необхідної пам’яті перед аналогами.

Характеристики нейронних мереж

Тестований мобільний пристрій	Модель нейронної мережі	Частота кадрів у секунду (FPS)	Час розпізнавання одного регіону (секунд)	Вага моделі (МБ)	Кількість класів об'єктів
Iphone X	YOLOv3-tiny	26–29	0.5–07	35	40
Iphone X	YOLOv3-416	2–3	5–6	250	80
Iphone X	YOLOv3-KD	24–26	0.5–0.6	21	2
Iphone 6s	YOLOv3-tiny	22–26	0.8	250	80
Iphone 6s	YOLOv3-416	2	3–4	35	40
Iphone 6s	YOLOv3-KD	20–24	0.7–0.8	21	2

Висновки

У роботі проаналізовано методи пошуку і розпізнавання об'єктів та показано доцільність побудови системи пошуку і розпізнавання об'єктів для мобільної платформи iOS в реальному часі. Запропоновано та досліджено метод функціональної адаптації алгоритму пошуку та розпізнавання об'єктів до особливостей відеозображень, із використанням коректувальних фільтрів.

Розроблено алгоритмічно-програмні засоби для вирішення задачі знаходження та оперативного розпізнавання об'єктів у режимі реального часу мовою Swift під мобільну платформу iOS.

На основі моделі Yolov3 створено модель згорткової нейронної мережі – Yolov3-KD. Наведено загальну структурну схему системи, схему алгоритму розпізнавання об'єктів та схему тренування моделі. Запропоновану ЗНМ верифіковано на двох навчених класах об'єктів.

Продемонстровано приклад роботи програми. Показано доцільність використаних засобів, враховуючи мінімізаційні та згладжувальні фільтри, використовуючи відповідні формули та програмні реалізації, а також квантизацію вагових коефіцієнтів. Запропонована ЗНМ здатна розпізнавати об'єкти в реальному часі.

Виконано порівняння запропонованої моделі ЗНМ із близькими аналогами за основними характеристиками, зокрема за величинами частоти опрацювання кадрів відеозображень. Для моделі YOLOv3-KD, розробленої спеціально під мобільні додатки, отримано доволі високу частоту кадрів: 24–26 кадрів у секунду, що цілком достатньо для роботи у режимі реального часу.

Список літератури

1. Wikipedia. Binary classification [Elektronnyj resurs] / Chicago 2019. Rezhym dostupu: https://en.wikipedia.org/wiki/Binary_classification.

2. Wikipedia. Support-vector machine [Elektronnyj resurs] / Chicago 2019. Rezhym dostupu: https://en.wikipedia.org/wiki/Support-vector_machine.

3. Wikipedia. Artificial neural network [Elektronnyj resurs] / Chicago 2019. Rezhym dostupu: https://en.wikipedia.org/wiki/Artificial_neural_network.

4. Wikipedia. Convolutional neural network [Elektronnyj resurs] / Chicago 2019. Rezhym dostupu: https://en.wikipedia.org/wiki/Convolutional_neural_network.

5-7. Rohith Gandhi. R-CNN, Fast R-CNN, Faster R-CNN, YOLO. Object Detection Algorithms [Elektronnyj resurs] / San-Francisco 2018 – Rezhym dostupu: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

8. Wikipedia. Keras [Elektronnyj resurs] / Chicago 2019. Rezhym dostupu: https://en.wikipedia.org/wiki/Keras.

9-10. Hao Gao. Understand Single Shot MultiBox Detector (SSD) and Implement It in Pytorch [Elektronnyj resurs] / San-Francisco 2018. Rezhym dostupu: https://medium.com/@smallfishbigsea/understand-ssd-and-implement-your-own-caa3232cd6ad.

11. Juan Garcia, Reza Bakhshandeh. Methods and systems for object recognition. [Elektronnyjresurs] / MenloPark2016-Rezhym dostupu: https://patents.google.com/patent/US948940.

12. Wikipedia. Google Translate [Elektronnyj resurs] / Chicago 2019. Rezhym dostupu: https://en.wikipedia.org/wiki/Google_Translate

METHODS FOR REAL-TIME OBJECT SEARCHING AND RECOGNIZING IN VIDEO IMAGES ON IOS MOBILE PLATFORM

D. Kushnir, Y. Paramud

Lviv Polytechnic National University,
Computer Engineering Department

The features of the most common methods and systems for searching and recognizing objects in video are explored. The research shows the feasibility of building search and recognition tools for the iOS platform in real time. The method of functional adaptation of the algorithm of search and recognition of objects to features of video is offered, which consists in processing of video image by smoothing and minimization filters, which reduces the time of search and recognition of objects. The block diagram and algorithm of system functioning were designed. Developed a program to solve the problem of finding and quickly recognizing objects in real time in Swift language on the iOS mobile platform. A convolutional neural network with YOLOv3 architecture was used along with framework for working with neural networks for mobile CoreML applications. A method of improving the performance of such a neural network is proposed, which is based on the quantization of the neural network weights and minimizes the model size and search time of its objects. The frequencies of image processing using the proposed means and models of neural networks of the type YOLOv3-tiny, YOLOv3-416 and our own model YOLOv3-KD are investigated. The possibility of functioning of the proposed funds in real time is provided.

Key words: object search time, object recognition time, video, mobile platform, convolutional neural network, real time.