КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

ЗАСТОСУВАННЯ ГЛИБИННОГО НАВЧАННЯ В ЗАДАЧАХ РОЗПІЗНАВАННЯ ОБРАЗІВ НА АЕРОЗНІМКАХ

Виконав студент 4-го курсу
Михайло ЛОГАЧОВ

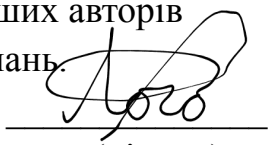
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Максим ВЕРЕС

_____

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент



(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«25» травня 2022 р.,
протокол № 10
Завідувач кафедри

Олександр ПРОВОТАР _____

(підпис)

Київ - 2022

РЕФЕРАТ

Обсяг роботи 44 сторінки, 36 використаних джерел, 12 рисунків.
Ключові слова: ГЛИБИННЕ НАВЧАННЯ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КОМП'ЮТЕРНИЙ ЗІР, АНАЛІЗ АЕРОФОТОЗНІМКІВ, ДИСТАНЦІЙНЕ ЗОНДУВАННЯ.

Об'єктом роботи є будова та навчання глибинних нейронних мереж для розпізнавання об'єктів на зображеннях. Предметом роботи є моделі нейронних мереж, здатних розв'язувати задачі локалізації та класифікації об'єктів.

Метою роботи є опис та порівняння широко застосовуваних архітектур глибинних нейронних мереж для розпізнавання штучних образів на аерофотознімках, та аналіз їх варіацій, що підходять для аналізу зображень в реальному часі.

Інструменти розробки: мова програмування Python, фреймворк побудови нейронних мереж Tensorflow, бібліотека комп'ютерного зору OpenCV.

Результатами роботи є треновані моделі, здатні для розпізнавання декількох класів штучних об'єктів та виділення їх положення на зображенні. В процесі роботи були досліджено сучасні підходи до задачі розпізнавання образів на основі повних згорткових глибинних мереж – Single Shot MultiBox Detector (SSD) та You Look Only Once (YOLO), запропоновано нові види мереж на основі розглянутих архітектур. Було відтворено експеримент з модифікацією SSD моделі, у якому вводяться додаткові проміжні шари та демонструються наслідки на загальний час тренування та вихідної точності в залежності від гіперпараметрів.

Областю застосування отриманих моделей є повністю чи частково автоматизовані апаратно-програмні системи дистанційного зондування Землі. За задумкою, система може бути використана для збору кількісної та якісної інформації про об'єкти на вказаній ділянці Земної поверхні, відтвореної на цифрових фотографіях для швидкого сповіщення та подальшого прийняття рішень щодо територіального планування. Результати, отримані в даній роботі,

можуть слугувати підґрунтям для подальших досліджень у напрямках комп’ютерного зору та дистанційного зондування.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ЗНН – Згорткова нейронна мережа;

ПВО – Перетин відносно об’єднання, оцінка якості локалізації об’єктів;

ШНМ – Штучна нейрона мережа;

FPN – Feature Pyramid Network, нейрона мережа, що комбінує вхідні дані між собою та просуває їх на вихід;

MSE – Mean Square Error, середньоквадратична помилка;

SSD – Single Shot Multi-Box Detector, архітектура нейронної мережі для виявлення рамок об’єктів та їх класів;

YOLO – You Look Only Once, Detector, архітектура нейронної мережі для швидкого виявлення рамок об’єктів та їх класів.

ВСТУП

Оцінка сучасного стану об’єкта дослідження. Історія дистанційного зондування налічує близько 60 років, і починається з створення засобів для зйомки. Разом з появленням нового джерела інформації виникла потреба і в автоматизації її аналізу. Після затяжного простою, лише в останні десятиріччя, з розвитком обчислювальних потужностей, у дослідників з’являтись інтерес до машинного навчання та штучних нейронних мереж у контексті систем автоматичного прийняття рішень [1]. Нові технології відкрили можливості як для масового збору вхідних даних, так і побудови складних мереж глибинного навчання, що вміють розв’язувати низку задач: регресія, класифікація, кластеризація і оптимізація. Також такі моделі здатні до самостійного навчанню особливостям вхідних даних, що є значною перевагою у порівнянні з класичними статистичними методами аналізу. Незважаючи на плюси, сучасні технології все ще не дозволяють штучним мережам симулювати розумові процеси вищих істот, тобто у великих масштабах, тому мережі повинні бути сконфігуровані та навчені окремо під кожну задачу.

Під час дослідження зору у кішок, Девід Х. Хьюбел і Торстен Візель зробили висновок, що велика кількість нейронів у зоровій корі мозку ссавців реагують на зорові подразники в деякій невеликій області поля зору, особливо на елементарні геометричні візерунки: лінії в різних направленнях та точки [2]. Більш того, сприймальні нейрони можуть перекривати один одного, тим самим ускладнюючи потенційні сприйнятливі форми. Як наслідок згорткові нейронні мережі (ЗНН) були створені для покращення знаходження графічних форм та елементів на малюнках, та отримали розповсюдження в області комп’ютерного зору. Так, модель LeNet-5, розроблена у 1998 році Яном Ле Куном [3], була прийнята у банковій сфері для розпізнавання написаних чисел, що підтвердило ефективність ЗНН. У сучасних підходах аналізу зображень зазвичай використовують великі комбінації таких мереж [4][5], разом з звичайними моделями глибинного навчання, що дозволяє одночасно врахувати

різні форми та кольори об’єктів, і отримати на виході бажаний формат, в залежності від типу задачі. Проте, як і у звичайному глибинному навчанні, при використанні ЗНН повинні бути враховані можливі проблеми навчання.

Актуальність роботи та підстави для її виконання. Існуючі підходи дозволяють розв’язувати задачі локалізації та класифікації об’єктів з точністю, що перевищує 90% [6]. Але, беручи до уваги, що вхідні дані представляють собою фотографії Земної поверхні, де об’єкти зображуються у невеликому розмірі, та розташовані або в угрупованнях, що є характерним для міської місцевості, або дуже віддалено, що характерно для зображень водної поверхні та сільської місцевості, відкриваються можливості для оптимізацій існуючих алгоритмів [7]. Тому у цій роботі розглядаються модифікації моделі Single Shot Multi-Box Detector [8] для виявлення образів, що потребують менше часу на навчання та аналіз фотографій.

Мета й завдання роботи. Мета роботи полягає у розробці архітектури з використанням спрощеної Feature Pyramid Network (SFPN) [9] в якості проміжного шару у мережі SSD, для досягнення близької швидкодії у порівнянні з моделлю YOLO [10], без значних втрат в точності при виділенні кораблів та літаків з аерофотографій. Одним з поставлених задач є і збір інформації про існуючі моделі, що використовують ЗНН та глибинні нейронні мережі, систематизація знань про них, опис будови та процесу відлагодження моделей SSD, SSD з Weighted Bi-Directional Feature Pyramid Network (BiFPN) [11], та YOLO, під сприйняття аерофотографій та порівняння результатів цих підходів, виведення значень впливу гіперпараметрів на остаточний результат.

Об’єкт, методи й засоби дослідження або розроблення. Об’єктом дослідження виступають глибинні нейронні мережі та їх варіації для побудови моделей комп’ютерного зору.

Об’єкт розробки – проведення дослідження ефективності роботи моделі з застосуванням SFPN в архітектурі SSD при аналізі зображень.

Основою середовища стала операційна система з відкритим вихідним кодом Fedora 35 на ядрі Linux 5.17, мова загального призначення Python 3.8,

інтерактивна обчислювальна платформа з веб-інтерфейсом Jupyter Notebook, веб-сервіс для віддалених обчислень Google Collab, бібліотека з відкритим вихідним кодом для обробки зображень OpenCV. У якості фреймворку глибокого навчання було обрано Tensorflow 2.7, що підтримує Keras Functional API. Основним інструментом розробки була обрана інтегрована середа розробки Visual Studio Code від Microsoft з відкритим вихідним кодом.

Можливі сфери застосування. Результат роботи рекомендується використовувати при проведенні наукових досліджень, у навчальному процесі, практичній діяльності фахівців у сфері машинного навчання та комп'ютерного зору. У роботі розкривається тематика модифікації архітектур під конкретну прикладну задачу, основні проблеми та їх розв’язки, що виникають під час використання нових даних. Структура отриманих моделей може бути перенесена на інші фреймворки та бібліотеки глибинного навчання і застосовуватись для військових і громадських цілей у вбудованих та GIS системах, таких як: розвідувальні дрони та супутникові установки. Ефективне виявлення штучних образів дозволяє застосовувати моделі у системах реального часу, що є критичним в задачах швидкого реагування, дозволяє зекономити час працівників, та зменшити ймовірність помилок через неуважність. Самі моделі здатні до навчання на будь-яких зображеннях заданого формату, що розширює простір класів об’єктів для розпізнавання.

Аналіз існуючих рішень. На ринку є програмні засоби, що дозволяють будувати системи машинного навчання і аналізу аерофотознімків, наприклад Nyckel. Продукти на зразок Nearmap показують хорошу точність, але не мають відкритого коду та часто беруть на себе відповідальність за збір даних для навчання, що може бути критичним при специфічних задачах або приватних розробках. Альтернативою є дослідницькі лабораторії та проекти, наприклад Oak Ridge National Laboratory або ArcGIS, що аналізують предметну область та грунтують свої висновки на її особливостях.

РОЗДІЛ 1. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

1.1 Основні поняття.

Штучні нейронні мережі є основою для глибинного навчання. Вперше вони були представлені у вигляді спрощеної моделі мозку, що складалася з комплексу поєднаних нейронів, нейрофізіологом Уорреном Мак Калохом і математиком Уолтером Питтсом в 1943 році [12].

Найпростішою варіацією ШНМ є пороговий логічний блок (ПЛБ) – функція виду:

$$y = \sigma(w^T x + b) \tag{1.1}$$

де y – вихідне значення мережі; x – вектор вхідних даних; w – вектор вагових коефіцієнтів; b – значення зсуву; σ – функція активації. Дана архітекрура не здатна відтворювати відносини між вхідними і вихідними значеннями у більшості випадків, тому Френком Розенблатом у 1958 була зпропонована ідея перцептрону – набору ПЛБ, або шаром нейронів. Багатошарові повнозв’язані моделі представляють собою послідовності перцептронів, нейрони яких на вхід приймають виходи усіх нейронів з попереднього шару. При цьому кінцевий вихід буде утворений враховуючи відповіді багатьох нейронів, що і дозволяє таким моделям навчатись і зберігати отриману інформацію. Нейротерапевт Дональд Хебб у своїй роботі присвяченій аналізу активності нейронів [13] вивів принцип, за яким моделі можуть навчатись: "Нейрони, які активуються разом, поєднуються разом" – звідси при отриманні схожих вхідних даних будуть активізуватись однакові нейрони, що і призводить до однакової відповіді.

Архітектура ШНМ складається з послідовності різних шарів, що беруть участь у перетворенні вхідних даних. Такі параметри, як: функція активації, кількість нейронів та шарів є гіперпараметрами і задаються при створенні моделі ШНМ.

Тренування ШНМ полягає в знаходженні оптимальних значень внутрішніх параметрів моделі, що оптимізовують функцію (1.2) втрат J, де y - вектор очікуваних значень а $\widehat{y}$ – вихід ШНМ на останньому шарі.

$$loss = J(y - \widehat{y}) \tag{1.2}$$

1.2 Функції активації.

На відміну від старих моделей, що базувались на пороговій функції, сучасні використовують різноманіття функцій активації. Головними вимогами для неї є велика область визначення (зазвичай усі дійсні числа), обмежена множина значень, простота обчислення і швидкість диференціювання. Одними з популярних [14] функцій є:

1. Step: $\sigma(x) = \{0: x < 0; 1: x \geq 0\}$
2. Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
3. Softmax: $\sigma(x) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$
4. Tanh: $\sigma(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
5. ReLU (rectified linear unit): $\sigma(x) = \{0: x < 0; x: x \geq 0\}$
    a. Leaky ReLU: $\sigma(x) = \{ax: x < 0; x: x \geq 0\}$
    b. ELU (exponential linear unit): $\sigma(x) = \{a(e^{x} - 1): x < 0; x: x \geq 0\}$

Вибір правильної функції залежить від типу вихідних даних, наприклад в останньому шарі часто [15] використовують варіації ReLU для отримання значень у великому діапазоні, та Softmax для отримання ймовірнісної оцінки при однокласовій класифікації, бо кожен вихідний нейрон повертає значення в діапазоні від нуля до одиниці, та сума всіх значень не перевищує одиницю.

1.3 Функції втрат.

В залежності від типу поставленої задачі ціль нейронної мережі може відрізнятись. Для власне моделі це означає різні функції втрат та цільове оптимізоване значення. В задачах регресії найбільш розповсюдженою є середньоквадратична помилка:

$$MSE(\widehat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \quad (1.3)$$

де $\widehat{y}$ – вихід нейронної мережі; y – цільове значення. Чим менше значення втрат, тим краще ШНМ пристосовується до вхідних даних.

В задачах кластеризації при заданій кількості кластерів k: $\{c_1, c_2, ..., c_n\}$ можна використовувати функцію (1.4).

$$L(x, c_1, c_2, ..., c_k) = \sum_{i=1}^{n} min_j ||x_i - c_j||^2 \quad (1.4)$$

В задачах класифікації часто представляють вихід мережі як вектор з нулів та одиниць [16], де одиниця на і-тому місці означає приналежність і-тому класу. У випадку однокласової проблеми, цей вектор ще називають one-hot vector. Щоб виміряти похибку між векторами, часто використовують дискретну перехресну ентропію (1.5) [17]. Важливими аспектами цієї функції є її диференційованість та утворення великих втрат при помилкових прогнозах, визначених з високою ймовірністю.

$$CrossEntropyLoss(\widehat{y}) = -\sum_{i=1}^{n}(y_i log(\widehat{y}_i) + (1 - y_i) log(1 - \widehat{y}_i)) \quad (1.5)$$

1.4 Тренування моделі.

Процес тренування або навчання мережі потребує три проходить крізь усі шари і розбивається на наступні кроки:

1. З вхідного набору даних обирається невелика вибірка і передається на вхід мережі декілька разів (епох) підряд.
2. Над кожним екземпляром даних шари нейронної мережі почергово виконують перетворення. Це відбувається шляхом обчислення значення функції активації над отриманими даними кожним нейроном.
3. Далі алгоритм знаходить вихідну помилку (втрати) мережі.
4. Алгоритм проходить по мережі в зворотному напрямку, враховуючи на кожному шарі часткові похідні втрат відносно всіх параметрів шару. Ланцюгове правило диференціювання функції допомагає розкласти похідні:

$$\frac{\partial J}{\partial \theta^L} = \frac{\partial J}{\partial y^L}\frac{\partial y^L}{\partial z^L}\frac{\partial z^L}{\partial \theta^L} = \frac{\partial J}{\partial y^{L-1}}\left(\frac{\partial z^L}{\partial y^L}\frac{\partial z^L}{\partial \theta^L}\sigma'(z^L)\right) \quad (1.6)$$

де L – номер шару; θ^L – параметри моделі, що змінюються; y^L – вихід L -того шару мережі; J – функція втрат; σ – функція активації; $z = f(\theta, x)$, або у випадку перцептронів $z = w^t x + b$ – латентна змінна. Алгоритм вимірює, скільки з цих втрат внесло кожне з'єднання в шарі нижче, знову ж таки, використовуючи правило ланцюга, працюючи назад поки алгоритм не досягне вхідного шару. Зворотний прохід ефективно вимірює градієнт помилки для всіх ваг з'єднань у мережі шляхом поширення градієнта помилки назад через мережу.
5. В кінці алгоритм виконує крок градієнтного спуску, щоб оновити всі ваги з'єднання в мережі, використовуючи градієнти помилок, які він обчислив.

Градієнтний спуск є методом оптимізації параметрів моделі у напрямку протилежному напрямку градієнта та пропорційно його значенню. Алгоритм повного градієнтного спуску (ПГС) можна представити у вигляді ітеративної функції:

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta J(\theta_i) \tag{1.7}$$

де θ_i – значення параметрів на і-тому кроці; $J(\theta)$ – функція втрат; η – величина кроку, або швидкість навчання. З формули можна зробити висновок, що цей алгоритм має лінійну швидкість збіжності, але повільну швидкість ітерації, обумовленою необхідністю порахувати значення градієнту для кожного нейрона для усіх вхідних значень, що лише для одного кроку означає пропускання через нейронну мережу увесь набір даних, обчислення помилки і коригування вагів для кожного екземпляра, лише після цього обчислення сумарної корекції вагів і застосування результатів. В глибинних ШНМ це може бути критичним місцем системи, що витрачає багато часу [18]. Щоб подолати цю проблему існує різноманітність модифікацій алгоритму градієнтного спуску [19], кожен з яких оптимізує той чи інший аспект.

Для зменшення кількості вхідних даних, що беруть участь в обчисленні градієнту використовують такі апроксимації:

1. Стохастичний градієнтний спуск (СГС) – грунтується на обчислені градієнту лише по одному випадково обраним на кожному кроці екземплярі. Звісно, напрямок зміни параметрів тоді не буде швидко просувати модель до глобального оптимума. Незважаючи на малі кроки при навчанні, збільшуючи кількості ітерацій, алгоритм буде наближатись до оптимума.
2. Міні-пакетний градієнтний спуск (МПГС) використовує випадкову вибірку з тренувальних даних, що дозволяє комбінувати переваги СГС та ПГС.

Для зменшення кроків для навчання використовуються припущення щодо наступного значення градієнту:

1. Оптимізація моменту: градієнт використовується в якості прискорення, а не швидкості, якщо проводити аналогії з рухом фізичних об’єктів. Це дозволяє плавно змінювати напрямок пошуку, та слабко залежати від аномалій у тестовій вибірці (1.8).

$$m_{i+1} = \beta m_i - \eta \nabla_\theta J(\theta_i)$$
$$\theta_{i+1} = \theta_i + m_i \quad (1.8)$$

2. Прискорення градієнта Нестерова. Якщо використовувати оптимізацію моменту, то наступне значення параметрів буде $\theta_i + m_i$. У 1983 році Нестеров запропонував використовувати нове знайдене значення параметрів в якості точки обрахування градієнту (1.10), тим самим пропускаючи один крок розрахунків.

$$m_{i+1} = \beta m_i - \eta \nabla_\theta J(\theta_i + \beta m_i)$$
$$\theta_{i+1} = \theta_i + m_i \quad (1.9)$$

3. Адаптивне передбачення моменту (Adam – adaptive moment estimation) основується на двох правилах: оптимізація моменту та припущенні, що ефективність спуску збільшується, якщо на початку крок буде великим, і, надалі, буде поступово зменшуватись, направляючи до точки оптимуму (1.10).

$$\theta_{i+1} = \theta_i + \frac{\eta \widehat{m}_i}{\sqrt{\hat{s}_i + \varepsilon}}$$
$$m_{i+1} = \beta_1 m_i - (1 - \beta_1)\nabla_\theta J(\theta_i)$$
$$s_{i+1} = \beta_2 s_i + (1 - \beta_2)\nabla_\theta J(\theta_i) \times \nabla_\theta J(\theta_i) \quad (1.10)$$
$$\widehat{m}_{i+1} = \frac{m_i}{1-\beta_1^i}$$
$$\hat{s}_{i+1} = \frac{s_i}{1-\beta_2^i}$$

В формулі (1.10) $\widehat{m}$ виконує роль адаптивного прискорення (momentum), а $\hat{s}$ – уповільнення (decay). Можна побачити, що при $\beta < 1$ зі збільшенням кількості ітерацій значення $\widehat{m}$ та $\hat{s}$ будуть рости (тому параметри

називаються експоненціально спадними середніми минулих градієнтів) і співвідноситись в пропорції оптимального значення кроку. Зазвичай їх значення встановлюють близькими до одиниці: рівними $\beta_1 = 0,9$ та $\beta_2 = 0,999$ – чим більше кроків потребує модель, тим більше значення. Проблема так званого поганого старту, коли початкові значення градієнту зіпсовують подальший рух тут вирішена зменшенням впливу градієнта пропорційно до параметрів β у $(1 - \beta)$ разів.

1.5 Налаштування моделі.

Серйозними проблемами, що виникають під час навчання є нестабільні градієнти. Вони стають все меншими і меншими коли алгоритм доходить до нижніх шарів [20]. В результаті оновлення градієнтний спуск залишає параметри з нижніх шарів майже незмінними, і навчання рідко завершується вдало. Це називається проблемою зникаючих градієнтів. У деяких випадках може відбутись вибухання градієнтів: вони стають настільки великими, що алгоритм починає розходиться. Іншими словами, нестабільні градієнти змінюють швидкість навчання в різних шарах.

При досліджені цієї задачі були виведені деякі оптимізації, що в обмеженому ступені нівелюють проблему, але в загальному випадку задача не є розв’язаною.

Першою причиною є неправильний вибір функції активації. Наприклад, функція активації ReLU має такий недолік: під час навчання деякі нейрони "вмирають", тобто вони перестають повертати значення, відмінні від нуля. У випадках великої швидкості навчання більшість нейронів мережі стають "мертвими": їх ваги змінюються таким чином, що значення z стають негативними для всіх екземплярів у навчальній вибірці. Коли це відбувається, мережа повертає однакові нульові відповіді, а градієнтний спуск більше не впливає на параметри, оскільки градієнт функції ReLU дорівнює нулю, коли її

вхід негативний. Альтернативою ReLU слугують протікаючі ReLU, що не залишають відповідь константою при негативних z.

Другою причиною є некорректна початкова ініціалізація вагів мережі. Якщо всім параметрам задані однакові значення, то градієнт всюди буде однаковий, і не буде призводити до навчання. Популярним підходом є He-initialization: вагам задаються випадкові значення з нормального розподілу на проміжку $(-\sqrt{\frac{3}{fan_{avg}}}, \sqrt{\frac{3}{fan_{avg}}})$ з медіаною нуль та $\sigma^2 = \frac{1}{fan_{in}}$, де fan_{avg} – середнє арифметичне з кількості виходів попереднього шару fan_{in} та входів наступного fan_{out}.

Третьою причиною може бути особливості вхідних даних, що призводять до проблемного градієнту під час навчання. Сергій Йоффе та Крістіан Чегеді у 2015 році запропонували [21] додавати спеціальній нормуючий шар нейронів між звичайними. Його задача полягає в обчисленні медіани μ та стандартного відхилення σ^2 з виходів попереднього шару та вирівнювання їх до стандартного розподілу з медіаною нуль та відхиленням рівним одиниці:

$$\hat{x} = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}} \tag{1.11}$$

Четверта причина проявляється в моделях з шарами, що складаються з багатьох нейронів – градієнтний спуск потребує багато часу, щоб їх врахувати. Тому в шарах під час тренування додають випадкове скидання вагів у деякої кількості нейронів. Такий підхід дозволяє перебирати комбінації нейронів майже незалежно один від одного, змушуючи їх пристосовуватись до нових особливостей даних. Викидання k відсотків вагів зменшує значення латентної змінної, тому ті нейрони, що залишилися збільшують значення своїх параметрів пропорційно до k . Після тренування модель вже не буде випадково себе поводити, тому всі ваги зменшуються у $1 + k$ разів.

П’ятою причиною є вибухаючий градієнт в окремих шарах мережі. Рішенням слугує обрізання градієнту під час зворотного поширення, щоб вони ніколи не перевищували деякий заданий поріг. Для того, щоб градієнт не змінив напрямок, наприклад у випадку обрізання вектору $\{1, 100\}$ до значень між $[-1, 1]$ утвориться вектор $\{1, 1\}$, вектор градієнта слід зменшувати на норму вектора.

РОЗДІЛ 2. ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ

2.1 Згорткові шари.

Оскільки згорткової нейронної мережі були розроблені для обробки зображень, тобто для двовимірних даних, надалі під згортковими шарами маються на увазі саме двовимірні структури. У контексті ЗНМ згортка — це лінійна операція, яка подібна до традиційної моделі перцептрону: скалярний добуток набору вагових коефіцієнтів, який називається ядром (kernel) або фільтром на вхідні дані. Фільтр називають вікном та його розмірність менша за вхідні дані, тому фільтр може застосовуватись до всіх частин вхідних даних. Щоб рівномірно розподілити застосування фільтра по зображенню, під час проходження крізь шар, його зсувають на величину s – додатковий гіперпараметер. Для нейронів біля кордонів вікно виходить за межі шару, в таких випадках або додаються нульові значення замість пропущених нейронів, або проблемні нейрони відкидаються, зменшуючи розмір вихідного шару.

Якщо фільтр призначений для виявлення деякої особливості у вхідних даних, то застосування цього фільтра по всьому вхідному зображенню дає йому можливість виявити її у будь-якому місці зображення. Оскільки фільтр застосовується кілька разів до одного вхідного масиву, результатом є теж двовимірний вихідний масив, що містить відфільтровані вихідні значення. Тому отриману структуру ще називають "картою ознак". Її значення можна направити на функції активації і отримати подібний шар до звичайних мереж. Під час навчання модель підбирає потрібне значення ядра, але кількість карт ознакою є гіперпараметром. При обробці зображень зазвичай на вхід передаються декілька кольорових компонент, тобто малюнки мають трьохвимірну просторову розмірність: довжина, ширина та кількість каналів. В реалізаціях популярних бібліотек машинного навчання згорткові шари дозволяють мати динамічну кількість бажаних карт ознак, для яких буде

шукатись окремий фільтр. Крім розділення на кольори, декілька фільтрів в одному шарі дозволяють виявляти різні геометричні особливості.

В склажних моделях кожен нейрон у згорткового шару пов'язаний з нейронами інших згорткових шарів, що дозволяє мережі зосередитися на невеликих низькорівневих функціях у одному прихованому шарі, а потім об'єднати їх у об’єкти вищого рівня в наступному прихованому шарі. Ця ієрархічна структура поширена в реальних зображеннях, це одна з причин, чому ЗНМ так добре працюють для розпізнавання зображень. Вихід нейронів обчислюється функцією:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',l'} \cdot w_{u,v,k',k}$$
$$i' = i \cdot s_h + u \qquad (2.1)$$
$$j' = j \cdot s_w + v$$

де $z_{i,j,k}$ – вихід нейрона з i-того рядка, j-того стовпчика, k-тої карти; $x_{i',j',k'}$ – вихід нейрона з попереднього шару та i'-того рядка, j'-того стовпчика, k'-тої карти; s_w та s_h - вертикальні та горизонтальні кроки зсуву; f_w та f_h – розмірності вікна; $f_{n'}$ – кількість карт в попередньому шарі; $w_{u,v,k',k}$ – вага зв’язку між k' і k-тою картою та нейроном з координатами u, v; b_k – зсув, аналогічній до звичайних мереж.

Проблема, що виникає в ЗНМ – велика кількість шуканих параметрів: припустимо, що з шар з k n на n карт зв'язується з шаром, у котрого r m на m карт, тоді кількість з’єднань буде $k \cdot r \cdot n^2 \cdot m^2$. Ця проблема розв’язується з двох сторін:

1. Кількості карт k та r можуть бути зменшиними, якщо застосувати новий згортковий шар не для мап n на n та m на m, а для "розвернутих" $k \times n$ та $r \times m$. Щоб гарантовано прибрати зайві карти, розмір фільтру встановлюється 1×1.

2. З іншого боку, розмірність вихідних мап можна ущільнити, застосувавши перетворення у менший простір. Найпростішим таким перетворенням є підвибірка, алгоритм якої схожий на звичайний фільтр, де його операція задається наперед та не змінюється.

2.2 Повні згорткові мережі.

Повністю згорткові мережі вперше були представлені в 2015 році Джонатаном Лонгом для розв’язку задачі семантичної сегментації [22]. Семантична сегментація заключається в знаходженні класу кожного пікселя зображення відповідно до клас об'єкта, якому він належить. Ключовою ідею підходу стала заміна щільних шарів у верхній частині ШНН згортковими шарами: при збільшенні кількості фільтрів, збільшується кількість вихідних карт, і при цьому кожен нейрон наступного шару буде отримувати значення активації попередніх з усіх карт.

Щоб перетворити класичний щільний шар у повний згортковий шар, кількість карт повинна дорівнювати кількості нейронів в щільному шарі, та розмір фільтру повинен дорівнювати розміру вхідних даних. Перевагами таких моделей є гнучкість у сприйнятті вхідних даних: оскільки мережа містить лише згорткові шари, при збільшенні розміру вхідних даних, кількість виходів буде змінюватись динамічно, через те, що матриця перетворення буде задіюватись і для нових даних. ПЗМ використовуються для побудови так званих модулів знаходження ознак (ЗО). Вони складаються з багатьох згорткових шарів різної розмірності, що дозволяє зберігати інформацію про геометричні форми у реальному світі. Через велику кількість параметрів їх тренування займає багато часу [23], але після навчання вони можуть використовуватись і в інших моделях.

Підхід перенесеного навчання (transfer learning) заключається в використанні навчений моделей в якості частин нових архітектур. Наприклад, якщо мережа розпізнає види тварин, то її модуль ЗО варто перевикористати у

моделі розпізнавання пород собак, бо незважаючи на якість нової мережі, вона буде як мінімум розпізнавати собак поміж інших об’єктів [24]. Хорошою практикою при навчанні такої мережі є багатокрокове тренування: спочатку "заморожуються" шари ЗО, тобто встановлюється заборона на зміну вагів, а потім розморожуються. Під час першого кроку друга частина моделі починає наближуватись до оптимуму, полагаючись на попередній натренерований шар. У другому етапі дозволяється зміна перенесеного модуля – підхід врятовує модуль ЗО від градієнту на недавно ініціалізованих нейронах.

РОЗДІЛ 3. РОЗПІЗНАВАННЯ ОБРАЗІВ НА ЗОБРАЖЕННЯХ

3.1 Локалізація об’єктів.

В найпростішому виді задача локалізації об’єкта на зображенні може бути виражена як задача регресії: щоб передбачити обмежувальну рамку (bounding box) навколо об'єкта, оцінюються горизонтальні та вертикальні координати центру об’єкта, а також його висота і ширина. На практиці можно масштабувати розмір зображення та відповідні координати об’єктів в простір $[0, 1]$, щоб не підставлятись під незахищені від вибухових градієнтів активаційні функції у шарах [25]. В таких сценаріях в якості функції втрат звичайна MSE не підходить, бо модель повинна навчатись виділяти територію з об’єктом а не підбирати координати. Тому часто втрати вираховують функцією перетину відносно об’єднання (ПВО):

$$IOU = \frac{S(Expected \cap Predicted)}{S(Expected \cup Predicted)} \tag{3.1}$$

де S – функція площі багатокутника.

У цього підходу є суттєвий недолік – він не виявляє декілька образів на зображенні, як мінімум через наявність лише чотирьох виходів. Першим покращенням, що приходить на думку, є розбиття вхідного зображення на невеликі клітини та обробка їх незалежно один від одного:

1. До виходів мережі додається коефіцієнт впевненості від нуля до одиниці (з використанням сигмоїдної активації та функцією втрат перехресної ентропії).
2. Якщо впевненість на сусідніх клітинах є однаково великою, рамки об’єднуються.

Очевидним недоліком є низька швидкість обробки: кількість проходів мережі напряму залежить від розміру вхідного зображення.

3.2 Архітектура YOLO.

У 2015 році Джозеф Редмонд запропонував архітектуру You Only Look Once (YOLO) [26], яка використовує особливість ПЗМ – за один прохід мережі можливо локалізувати об’єкти на всіх підклітинках, якщо розмір вікна на вихідному згортковому шарі буде дорівнювати розміру цієї клітинки.

Авторам вдалося отримати модель з розпізнаванням 150 кадрів в секунду, що дозволяє обробляти потокові відео в режимі реального часу. Додатково, YOLO глобально міркує про зображення, роблячи прогнози. На відміну від методів розсувного вікна та пропозицій регіону, YOLO аналізує усе зображення під час роботи, тому неявно кодує контекстну інформацію про класи, а також їх зовнішній вигляд. Архітектура YOLO складається з дев’яти основних шарів, кожен з яких містить декілька згорткових та збираючих. Як видно з рисунку (рис. 3.1), де зображується схема цієї моделі, кількість карт ознак збільшується з часом.

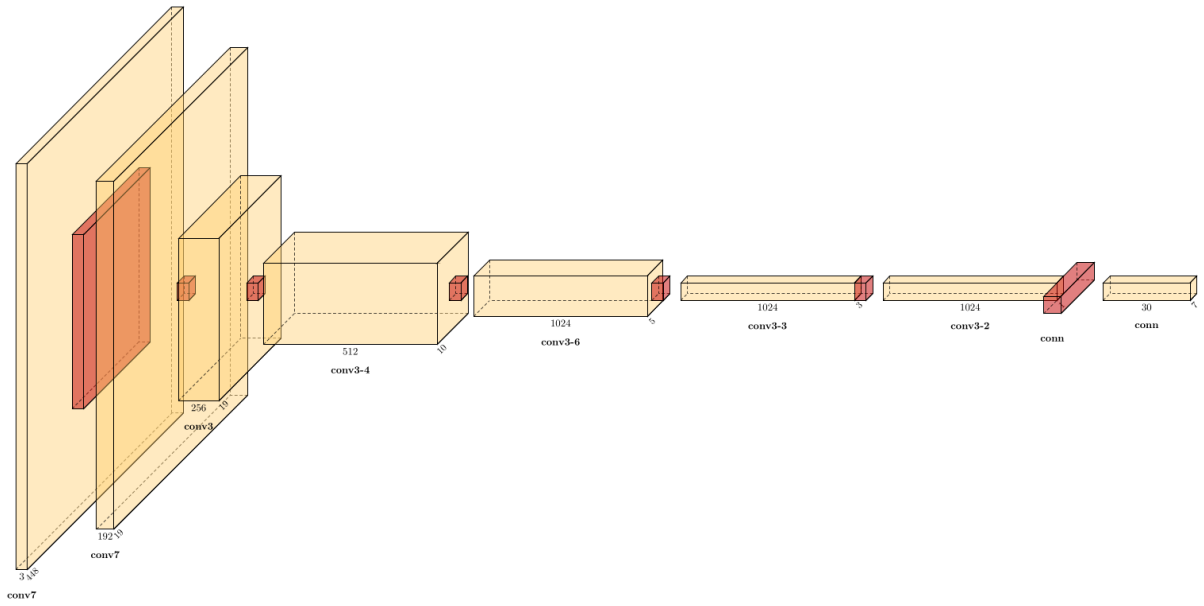


Рисунок 3.1 – Архітектура моделі YOLO

Оскільки YOLO дуже узагальнений алгоритм, він має меншу ймовірність отримати малу точність при застосуванні до нових доменів. Цей алгоритм розбиває вхідне зображення на сітку $S \times S$. Якщо центр об’єкта

потрапляє в клітинку сітки, ця клітинка відповідає за виявлення об'єкта. Кожна клітинка сітки передбачає обмежувальні рамки та показники впевненості для цих рамок. Ці показники відображають, наскільки впевнена модель у тому, що рамка містить об'єкт, а також наскільки точно вона вважає, який клас рамка прогнозує. Формула визначення впевненості для клітин:

$$P(Class \mid Location) = \frac{Pr(Class)*IOU}{P(Location)} \quad (3.2)$$

Оцінки демонстують як ймовірність появи цього класу в рамці, так і те, наскільки добре передбачена рамка відповідає об'єкту. Якщо в клітинці немає жодного об'єкта, показники впевненості мають бути нульовими. В іншому випадку оцінка довіри повинна дорівнювати ПВО між передбаченою та достовірно істиною рамкою (ground truth box). Тому сусідні клітини не прагнуть повністю охопити об'єкт, а лише його частини з кращою точністю. В якості функції втрат використовуються складна функція (3.3), вона штрафує за високу впевненість з неправильним передбаченням та штраф MSE від коренів значень висоти і ширини за неправильну рамку.

$$loss = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] + \quad (3.3)$$

$$\lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in C} (p_i(c) - \hat{p}_i(c))^2$$

В формулі (3.3) λ – коефіцієнти впливу різних видів похибок; x, y – координати центру рамки, $w,\ h$ – ширина і висота рамки, C – клас об'єкту; $p(c)$ – функція впевненості, $1_{i,j}^{obj}$ – функція, що дорівнює одиниці, коли клітинка i, j відповідає за розпізнавання рамки, інакше – нуль; аналогічно, функція 1_i^{obj} дорівнює одиниці коли в рамці i знаходиться і-тий клас об'єктів.

Недоліком MSE в даному випадку є велика кількість клітинок, що не містять жодного об’єкту. Показники впевненості цих комірок близькі до нуля, і їх кількість часто переважає кількість клітин з об’єктами, тому є ймовірність слабкого впливу градієнту на них. Щоб виправити це, збільшуються штрафи від прогнозування координат рамки і зменшується штрафи від передбачення надійності для клітин, які не містять об’єктів.

Незважаючи на переваги у швидкості, YOLO має строгі обмеження на мінімальний розмір образів – вони не повинні бути сильно меншими за клітинку сітки, оскільки кожна клітинка передбачає лише один клас [27]. В задачі розпізнавання зображень з повітряних та космічних апаратів це є серйозним недоліком, як далі буде показано. Ще один недолік заключається у виборі функції втрат, яка нормально сприймає невеликі помилки у великій рамці, в той час як невелика помилка в предсказаній точності для маленькій рамці має набагато більший вплив на фінальну оцінку.

3.3 Архітектура SSD.

Модель Single Shot MultiBox Detector (SSD) була опублікована в 2016 році Крістіаном Жегед і показала свою ефективність на багатьох стандартних наборах даних [28].

Як видно з діаграми (рис. 3.2), SSD складається з трьох основних компонент:

1. Виділення ознак (Feature extraction)
2. Виявлення рамок (Detection Heads)
3. Подавлення передбачень (Non-Maximum Suppression)

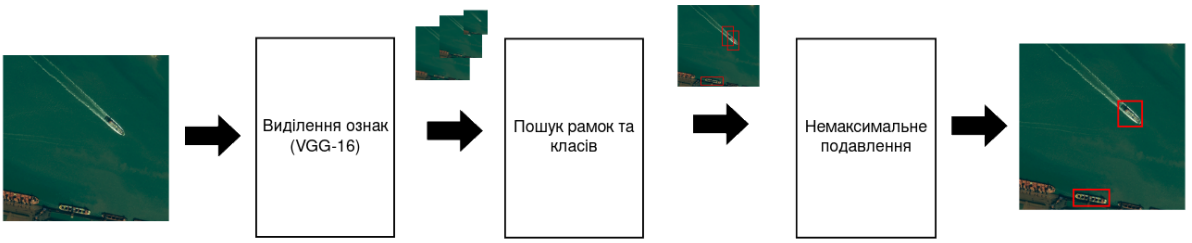
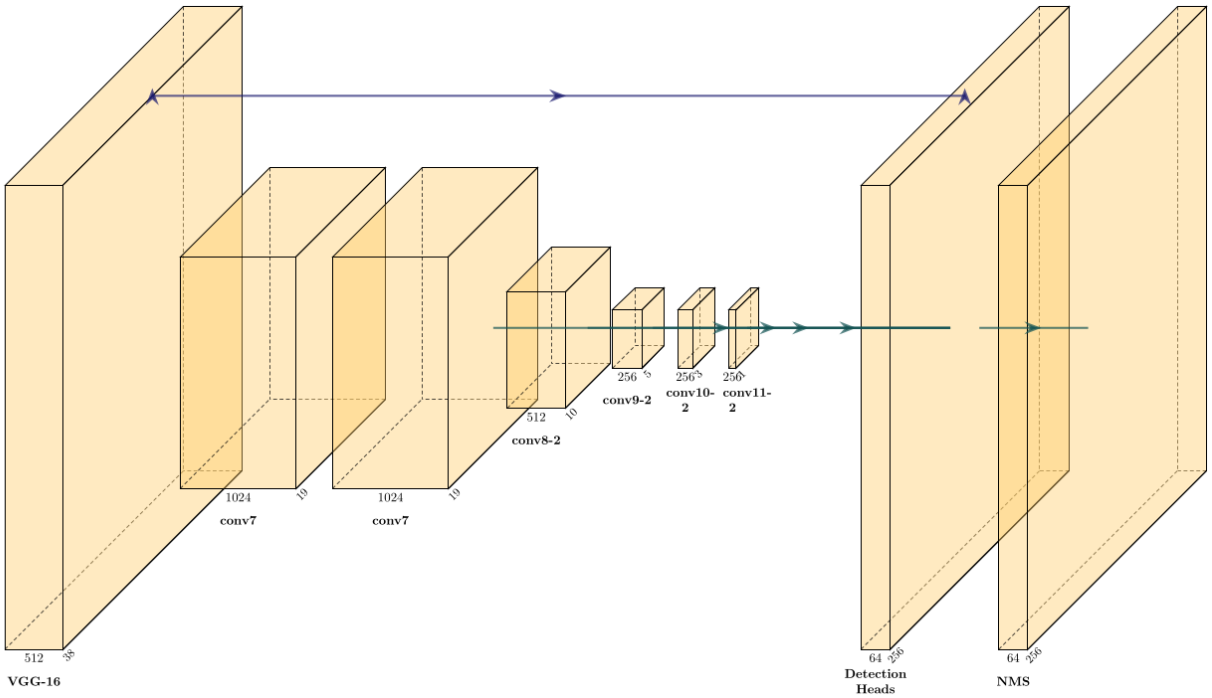


Рисунок 3.2 – Архітектура модулів моделі SSD

В якості шару виділення ознак використовується модель VGG-16, що складається з серії згорткових та підвибіркових шарів, її виходи подаються на момодель виявлення рамок.

Під час виявлення рамок до входу послідовно застосовуються шість згорткових та повнозв'язних шарів з різним вікном (4×4 з кроком 3, 7×7, 6×6, 9×9, 10×10, 11×11 з кроком 2). Отримані карти (multi-scale feature maps – MSFM) містять інформацію про особливості об'єктів різних



масштабів, що дозволяє ефективно їх знаходити (рис. 3.3).

Рисунок 3.3 – Архітектура модуля виявлення ознак

Для кожної клітини з карт перевіряється ефективність локалізації за допомогою рамок різної форми, що називаються default boundary boxes. На відміну від YOLO, яка сама визначає охоплюючі рамки, SSD потребує задання співвідношення сторін та масштабу, хоча останній можливо змінювати в залежності від номеру шару: (3.4), (3.5).

$$width = scale \cdot \sqrt{ratio} \tag{3.4}$$

$$height = \frac{scale}{\sqrt{ratio}} \tag{3.5}$$

Додатково до вказаних пропорцій за замовчуванням додаються рамки з

$$scale = \sqrt{scale_l \cdot scale_{l+1}} \tag{3.6}$$

де l – номер шару, та з $ratio$, що дорівнює одиниці.

Серед можливих рамок обирається одна з найбільшим значенням ПВО, при цьому рамки з ПВО менше 0,5 не розглядаються. Після успішного знаходження обмежуючих рамок для клітинок в них застосовується згортковий шар з невеликими вікнами у кількості рівній кількості класів, враховуючи порожній, що визначає очікуваний клас зображення.

3.4 Мережі ознак.

Мережа подавленя передбачень отримує список рамок та класів, і поєднує або видаляє близькі рамки, що повинні належати до одного об’єкта. Очевидно, що чим більше розмірність карти, тим менші об’єкти можна на ній знайти [29]. Це правило працює і в обернену сторону – великі об’єкти стають видимими лише при сильному стисненні. В SSD верхні карти ознак не отримують інформацію про глобальні особливості даних, тому був запропонований підхід пірамідальної мережі характеристик (FPN) [30], який за допомогою проміжних шарів передає інформацію назад по шарам (які виглядають як піраміда) (рис. 3.4).

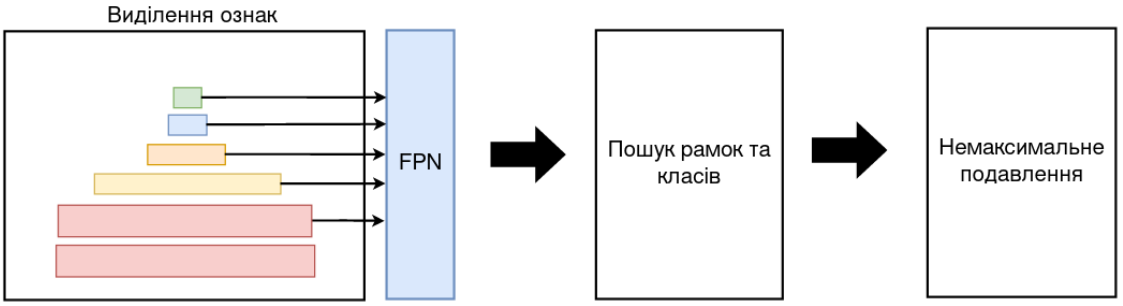


Рисунок 3.4 – Архітектура моделі SSD з проміжним модулем пірамідальної мережі ознак

Потік даних FPN складається з шляхів знизу вгору та зверху вниз. Шлях знизу вгору представляє собою класичну згорткову мережу для вилучення ознак. Піднімаючись, просторова роздільність зменшується, при цьому при виявленні більш високорівневих структур семантичне значення для кожного шару збільшується. Нижні шари мають високу роздільну здатність, але семантична цінність недостатньо висока, щоб виправдати їх використання, оскільки значну роль грає уповільнення моделі. Таким чином, звичайний SSD використовує лише верхні шари для виявлення об'єктів і, отже, працює набагато гірше для невеликих розмірів.

Якщо змінити джерело входу для наступного модуля з виділення ознак на мережу, створену FPN, нові карти ознаки будуть більш насиченими та містити чіткі ознаки образів. Нові карти утворюються шляхом реконструкції існуючих згори донизу, тобто використовуючи верхні шари як основу.

Хоча реконструйовані шари є семантично сильними, але координати та положення об’єктів не зберігаються після всієї процедури зниження та підвищення розмірності. Зазвичай додатково додаються бічні (сторонні) зв’язки між новими шарами та відповідними картами ознак, щоб допомогти модулю пошуку рамок класів краще передбачити розташування. FPN також діє як проміжна мережа, і спрощує основну мережу локалізації.

Шлях знизу вгору використовує подібну до ResNet мережу виділення ознак. Він складається з багатьох комбінованих згорткових шарів кожен з яких, в свою чергу, містить декілька простих шарів згортки та об'єднання. При русі вгору, розмірність зменшується на половину.

Шлях зверху вниз використову одиничний згортковий фільтр, щоб зменшити глибину вхідних шарів, додатково застосовуючи шар оберненої згортки для підвищення розмірності вдвічі, при цьому нові пікселі зображення приймають усереднене значення їх сусідів. Потім отримана карта додається до старої, тим самим фіксуючи початкові ознаки. Для того, щоб не сповільнювати процес обробки, на першому (останьому в черзі) шарі алгоритм зупиняється. Такий підхід, звісно, порушує логіку міркувань при проектуванні алгоритму, але дозволяє зекономити дисковий (графічний) простір та не сповільнювати процес обробки.

Однією з варіацій такої мережі є двонаправлена піраміда виділення ознак (BiFPN) [31], приблизна архітектура якої зображена на рис. 3.5: вона ефективно виконує завдання двонаправленої передачі параметрів, що в результаті покращує загальну точність моделі.

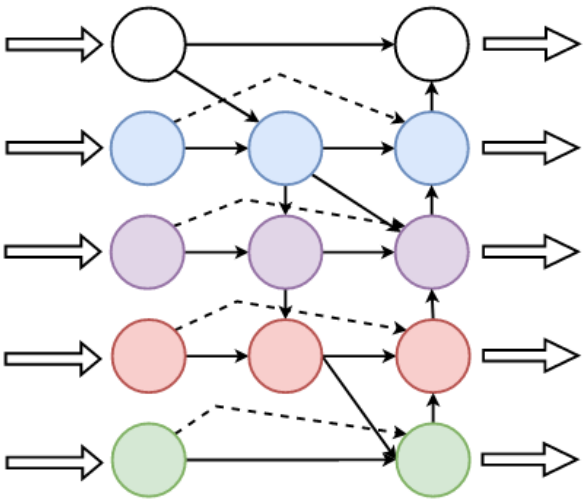


Рисунок 3.5 – Архітектура мережі BiFPN. Товстими стрілками позначені входи і виходи мережі, тонкими – операції передачі, збільшення та зменшення карт. Проміжний шар є результатом сумування вхідних до нього карт

Особливістю BiFPN у порівнянні з прямою передачею заключається у її модульності: Кожен наступний шар отримує пряму залежність від одного попереднього та проміжну залежність від усіх шарів до нього. Додатково, мережі BiFPN можна з'єднувати у послідовності, поступово зменшуючи вплив початкової піраміди ознак.

РОЗДІЛ 4. ПОБУДОВА МОДЕЛІ ДЛЯ ЗНАХОДЖЕННЯ ОБРАЗІВ

4.1 Постановка проблеми.

Моделі YOLO-типу досягають хороших результатів як з швидкості роботи, так і в точності знаходження та класифікації образів на фотографіях. Змінюючи щільність розбиття зображення, можна переконфігурувати модель на кращу точність в обмін на швидкодію. На жаль, зміст даних, отриманих з повітряних та космічних апаратів відрізняються від фотографій, зроблених на поверхні. Через велику висоту зйомки та перспективу, об’єктив камери захоплює велику площу, тому всі об’єкти зображуються в малому масштабі [32]. Для тіл, розмір яких перевищує десятки метрів це не є серйозною проблемою, в той час як малі тіла можуть повністю зникнути з зображення через алгоритми обробки та стиснення даних.

Як відомо з попереднього розділу, швидкодія YOLO буде падати, якщо спробувати охопити нею скупчення малих об’єктів. Альтернативою може слугувати моделі SSD, підкріплені пірамідальною мережею характеристик. Для них трапляється інша ситуація – висока точність при меншій за YOLO швидкодії. Одним з можливих місць до модифікацій і є ця мережа характеристик. Враховуючи, що згорткові шари на початку модуля знаходження рамок зберігають інформацію якраз про невеликі образи, та шари наприкінці можуть дати відповідь на питання класу території, що знімається (урбанізована зона, водна поверхня, поле, тощо), можна побудувати мережу, що буде витрачати менше з’єднань, і параметрів в порівнянні з BiFPN.

В якості задачі обрано мультикласову проблему виявлення розташування літаків та кораблів на фото, шляхом виділення охоплюючих рамок. В цьому випадку очікується, що бо модель буде вчитись добре ідентифікувати як випадки згрупованих тіл , наприклад у портах і аеропортах, так і віддалених.

4.2 Обробка вхідних даних.

Обробка вхідних даних зазвичай займає найбільшу кількість часу. При локалізації об’єкта треба знайти координати рамки, що охоплює кожен об’єкт на кожному фото. В цій роботі для тренування моделей використовується відкрита частина FAIR1M [33] датасету, що пропонує близько 1700 зображень 28-ми класів об’єктів.

Ці класи включають різні види машин, літаків, кораблів, а також будівель. Кожне зображення є квадратним з розширенням від 500 до 3000 пікселів, та містить хоча б один класифікуємий об’єкт. Через високу якість фотографій додаткова обробка не потрібна, але в загальному випадку варто додати фільтри згладжування, різкості або контрастності.

Моделі SSD так і YOLO чутливі до змін форми образів [34], тобто вони можуть не розпізнати вивчений образ, якщо той зустрівся у перевернутому положенні. Тому вхідний датасет був штучно розширений обробленими зображеннями: з ймовірністю p=0,1 до вхідних даних застосовувались операції масштабування на випадкову величину до 2-х, повороту не більше, ніж 3 градусів – щоб запобігти витрачанню очікуваної рамки, віддзеркалення, заміни кольорів та вирізка випадкового фрагмента. Датасет містить зображення об’єктів не в однаковій кількості, і в навчальній вибірці, котра обрана як 30% датасету, з високою ймовірністю може не опинитись класів, що рідко зустрічаються. Для включення усіх класів в тій же пропорції, що і вхідному датасеті вибірки формувались функціями бібліотеки Tensorflow.

4.3 Опис підходу.

Модель в даній роботі грунтується на використанні тренованої моделі VGG-16 на датасеті Pascal VOC в якості модуля виділення ознак, при цьому

для навчання застосовується алгоритм transfer learning, з порогом заморозки в п’ятдесяти епох.

Так само, як і в класичній SSD архітектурі, новостворена включає в себе шість згорткових шарів для розподілу характеристик на різні масштаби. Виходи з кожного шару передаються на згорткові шари-класифікатори та регресори, кожен з яких представляється 4 ма фільтрами 3×3 для знаходження положення, та n (кількість класів) фільтрами 3×3 для передбачення класу. Кількість класів рівне трьом: літаки, кораблі та фон (background). Для кожного оброблюваного шару задане значення масштабу рамки аналогічно до моделі SSD з MS COCO датасетом, але з доданим на початок масштабу в 0,03 разів. Пропорції використовувались ті ж самі, що і в оригінальному SSD. Для того, щоб випадковим чином рамки передбачень не зливаються в одну велику, для кожного шару встановлено мінімальне порогове значення між центрами рамок. В деяких зображеннях об’єкти були розташовані близько до границі, і алгоритм міг видати рамку, що заходить за межі зображення. Щоб не допустити цей випадок, центрам рамок не дозволялося розташовуватись за центрами клітинок для граничних територій.

Під час тестування було виведено, що стохастичний градієнтний спуск не сходиться до розв’язку, тому був обраний оптимізатор Adam з конфігурацією за замовчуванням. Одним із оптимізацій пошуку став вибір динамічної функції швидкості навчання: перші 30 епох мали n=0,001, надалі зменшуючись у 10 разів кожні 30 епох. Навчання відбувалось на серверах Google Collab з підтримкою CUDA, тому розмір batch встановлювався виходячи з обсягу доступної пам’яті у відеокарті, і рівним восьми. Кількість ітерації в епосі обиралось такою, щоб майже рівномірно поділити весь датасет для тренування за епоху:

$$N_{iter} = \frac{N_{train}\left(1 + \sum_{i=1}^{N_{augment}} P_i\right)}{batch\ size} \tag{4.1}$$

де $N_{augment}$ – кількість операцій розширення датасету; P_i – ймовірність утворення нового зображення завдяки зміні існуючого.

Архітектура мережі SSD включає мережу пірамідального поширення характеристик BiFPN з трьома блоками або з трьома блоками запропонованої спрощеної мережі (SFPN), схема якої наведена на рис. 4.1.

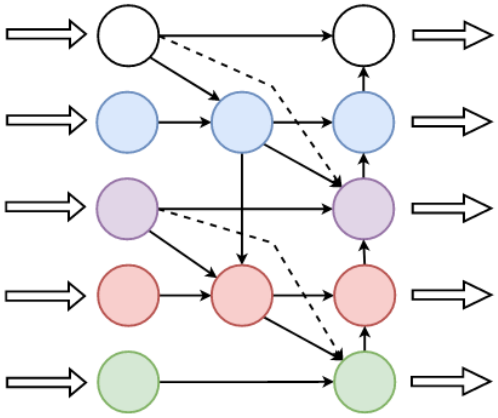


Рисунок 4.1 – Архітектура мережі SFPN

Спрощенна модель робить вплив на перший шар менше залежним від попередніх, проте середній шар поєднує сильний вплив найбільшого при підйомі вгору та від наймешного при спуску донизу.

В якості моделі YOLO була використана імплементація на Keras з Tensorflow backend [35] з значеннями за замовчуванням. Критерієм зупинки навчання було мале значення втрат (менше 1 для перехресної ентропії) або перевищення кількості епох (більше 1000).

4.4 Оцінка моделі.

В результаті тренування для задачі мультикласифікації всі моделі показують погані результати: впевненість у виборі не перевищує 0,2, тому допускає помилкові передбачення. Як було проаналізовано на датасеті Udacity Self Driving Car Object Detection, модель працює корректно для декілько міток. Однією з причин є невеликий розмір датасету та наявність некоректних даних.

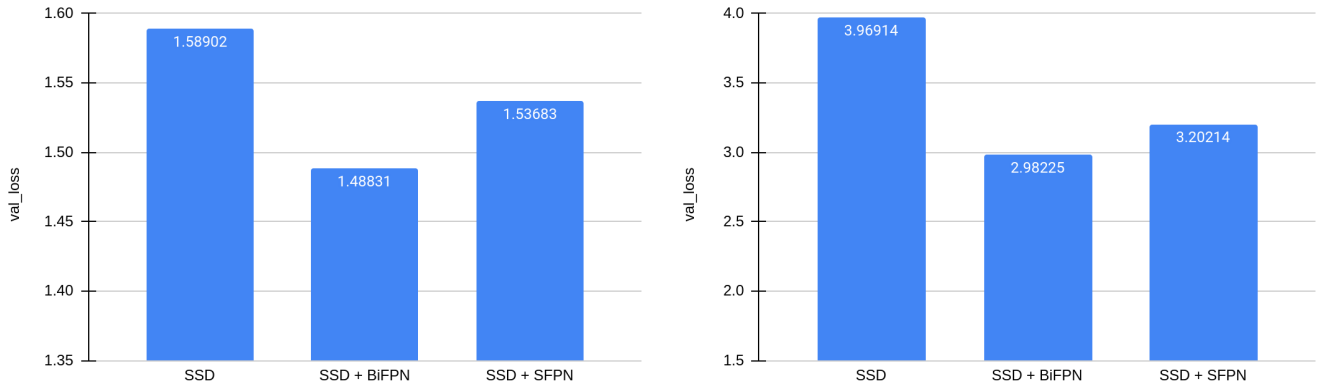
Щоб виправити цю проблему був використаний підхід голосування – дві мережі що розпізнають по одному класу дають свої оцінки зображенню, і в кінці обирається варіант з найбільшим значенням впевненості. Звісно, замість одного проходу модель робить один для кожного класу, але це є лише обмеженям навчальної вибірки, тому надалі будуть розглядатись моделі для передбачення лише одного класу.

Під час тестування модель показала високі значення впевненості на конкретних об’єктах та корректно знаходила їх рамки. Більш того, модель навчилися знаходити аномалії в датасеті (рис. 4.2).

(а) *(б)*

Рисунок 4.2 – рисунок (а) зображає очкувані результати, рисунок (б) – вихід натренерованої моделі

Точність вимірювалася функцією втрат, враховуючи значення впевненості, ПВО рамок, та знайденим класом. Модель YOLO використовує різницю коренів пропорцій рамки, замість ПВО у функції втрат, тому не враховувалась. SSD з SFPN показує точність на 3% і 17% кращу за SSD при локалізації рамками (рис. 4.3).



(а) *(б)*

Рисунок 4.3 – Середнє значення функції втрат на валідаційний вибірці, (а) – для фото літаків, (б) – для фото кораблів

Узагальнена точність класифікації враховувала лише коректність співпадіння класами. Модель SSD з спрощеною архітектурою поширення ознак розпізнає об’єкти на 6% краще за звичайну SSD. В той же час, YOLO погано справляється з локалізацією об’єктів, особливо – кораблів біля портів і літаків у аеропортах (рис. 4.4).

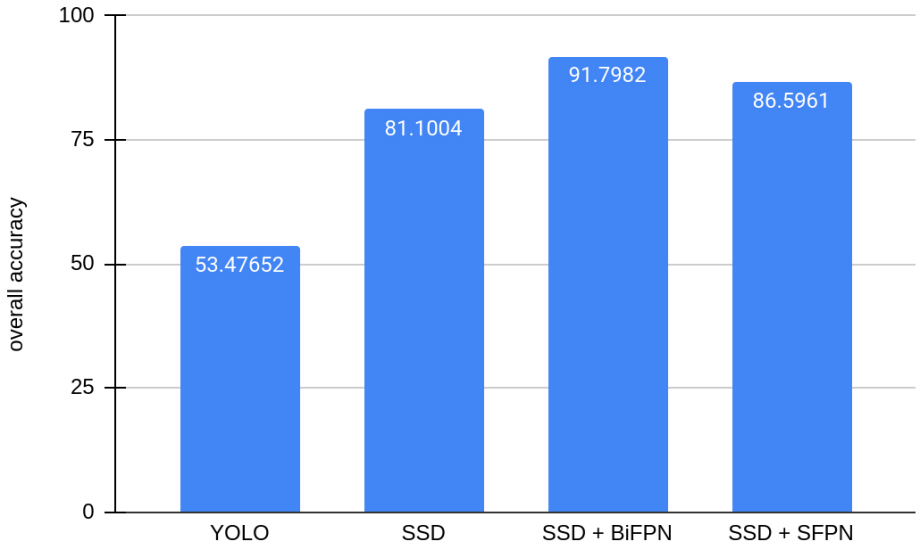


Рисунок 4.4 – Середнє значення точності класифікації

Швидкодія замріялась в середінй кількості оброблених зображень на секунду (рис. 4.5). Це значення є частотою обробки, для отримання часу аналізу однієї картинки треба знайти обернену оцінку.

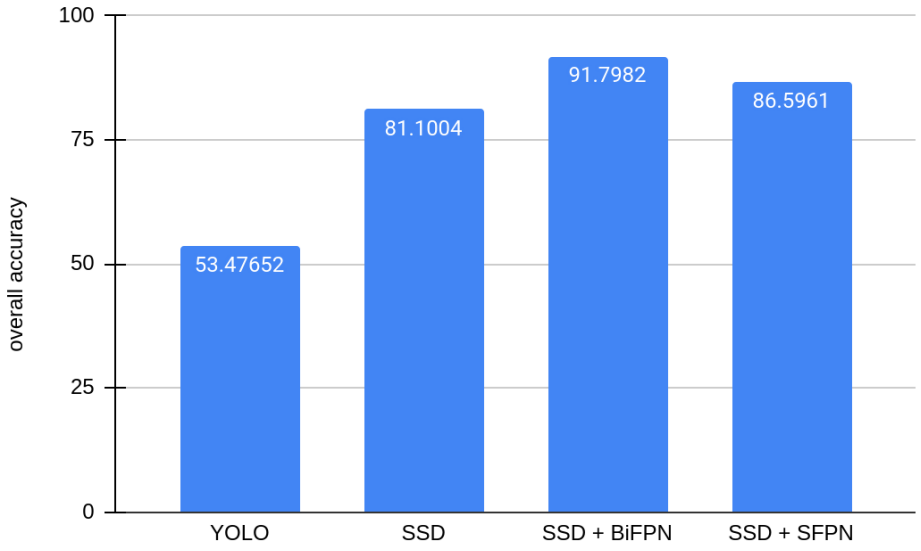
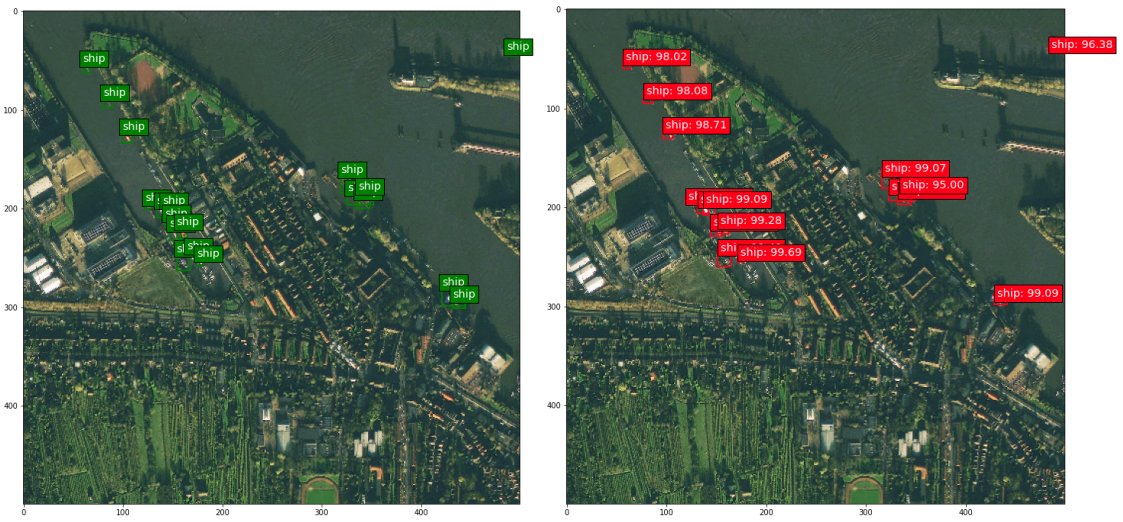


Рисунок 4.5 – Середня кількість кадрів на секунду при використанні моделей

SSD разом з SFPN показує трохи гірші значення точності, але при цьому піднімаючи кількість оброблених кадрів на секунду.

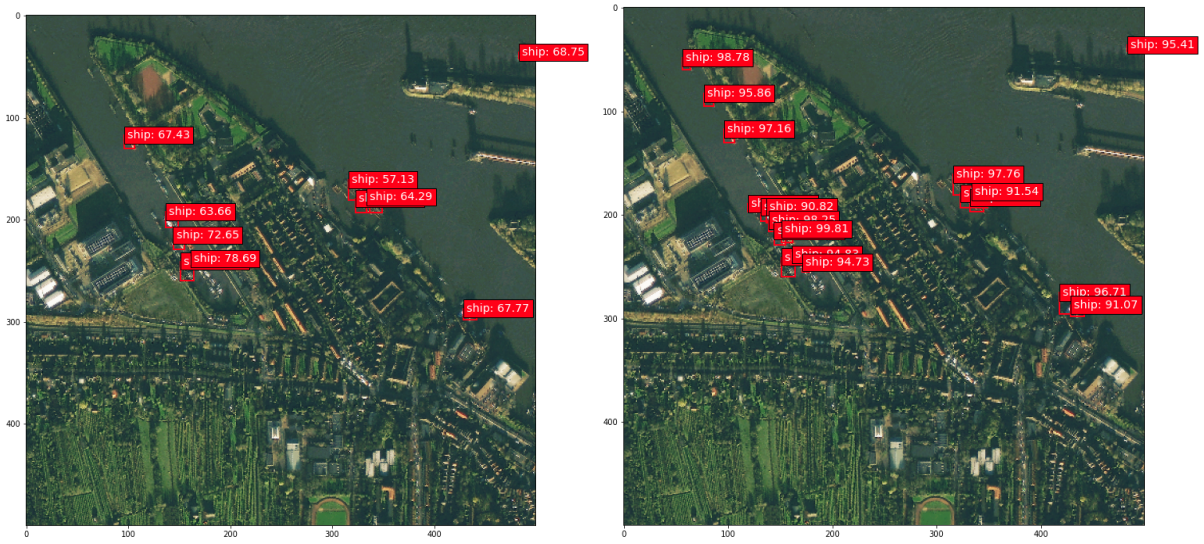
Незважаючи на повільну роботу SSD з BiFPN, разом з SFPN модель показує на 7% кращі результати. Значення частоти, звісно, є невеликим, у порівнянні з YOLO та класичним SSD, проте є достатньою для обробки потокового відео. На основі цього можливо зробити припущення, що аналогічно до проблеми влучності та повноті, існує мінімальна архітектура поширення ознак, що дає найкраще пришвидшення при хорошій точності.



(а) *(б)*

Рисунок 4.6 – Оригінальні об’єкти на зображенні (а). Об’єкти, виявлені моделлю SSD з SPFN (б)

На відміну від YOLO, отримана модель корректно опрацьовує випадки скупчення багатьох об’єктів (рис. 4.7) одного та різних класів та показує усереднені значення між двома існуючими варіаціями, тому може використовуватись в геоінформаційних системах та пристроях, що не мають достатньо обчислювальних ресурсів.



(а) *(б)*

Рисунок 4.7 – Об’єкти, виявлені моделлю YOLO (а). Об’єкти, виявлені моделлю SSD з SPFN (б)

4.5 Можливі покращення.

По-перше, кроком в сторону збільшення точності є розширення датасету більшим набором зображень з якісними позначками, оскільки, як було показано, штучного розширення не вистачає. По-друге, у кожної фотографії, знятої з неба є параметр висоти та нахилу. Якщо додати їх як аргументи напряму до Detection Heads, модель буде пристосовуватись брати на увагу образи на масштабованих згорткових шарах в залежності від обчисленого реального розміру об’єктів. По-третє, деякі джерела зйомки можуть надавати додаткові зображення, отримані спеціальними сенсорами. Наприклад, супутник Sentinel-5 отримує фотографії у різному світловому (частотному) діапазоні: від інфрачервоного до ультрафіолетового [36]. Викоистовуючи ці дані можна здогадуватись не тільки про колір, а і про тип відзнятого матеріалу, його щільності та пропускної здібності. Шар виявлення особливостей може отримувати ці дані та краще передбачати об’єкти: наприклад, кольоровий прямокутник несе менше інформації, ніж металевий кольоровий прямокутник на грунті.

ВИСНОВКИ

В даній роботі були описані теоретичні основи глибокого навчання та згорткових нейронних мереж. Окремо були розглянуті популярні архітектури мереж для розпізнавання об’єктів: класичний підхід, YOLO, SSD. Були описані переваги та недоліки цих підходів, і також вказані потенційні місця для покращення. Зроблено та перевірено припущення щодо доцільності використання власнорозробленої спрощеної пірамідальної мережі ознак в архітектурі SSD при розпізнаванні малих об’єктів. Для цього були виміряні та порівняні показники швидкодія та точність, на які користувачі звертають увагу при виборі моделі для своїх потреб.

Поставлена задача розпізнавання образів на аерофотознімках та виділені її особливості в контексті розглянутих моделей. Результати роботи демонструються у вигляді простих для аналізу графіків та малюнків. Запропонований у цій роботі підхід показує свою корисність, коли вхідні зображення мають великий масштаб. Додаткові питання, підняті у роботі є підставою для подальших досліджень в цій області.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aurélien Géron "Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow" SBN 978-1-492-03264-9, Second Edition, Published by O'Reilly Media, Inc 2019; г. 1, ст. 15.
2. Lienhard, Dina A., "David H. Hubel and Torsten N. Wiesel's Research on Optical Development in Kittens" ISSN: 1940-5030, Embryo Project Encyclopedia, 2017, (Препринт http://embryo.asu.edu/handle/10776/12995).
3. LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. "Backpropagation Applied to Handwritten Zip Code Recognition". Neural Computation, 1989, г. 1, ст. 541–551, (Препринт https://doi.org/10.1162%2Fneco.1989.1.4.541).
4. Song, Taeyong, Sunok Kim, SungTai Kim, Jaeseok Lee, and Kwanghoon Sohn. "Context-Preserving Instance-Level Augmentation and Deformable Convolution Networks for SAR Ship Detection" In 2022 IEEE Radar Conference (RadarConf22), 2022, ст. 1-6. IEEE, (Препринт https://arxiv.org/abs/2202.06513).
5. Chen, Xin, Qingtao Tang, Ke Hu, Yue Xu, Shihang Qiu, Jia Cheng, and Jun Lei. "Hybrid CNN Based Attention with Category Prior for User Image Behavior Modeling", 2021, (Препринт https://arxiv.org/abs/2205.02711).
6. Zhao, Mingpeng, Hanhui Li, Ruiqi Li, Ying Li, Xiaonan Luo, Tin Chiu Li, Tin Lap Lee, Wen Jun Wang, and David Yiu Leung Chan. "Automated and precise recognition of human zygote cytoplasm: A robust image-segmentation system based on a convolutional neural network" Biomedical Signal Processing and Control в. 67, 2021, (Препринт https://doi.org/10.1016/j.bspc.2021.102551).
7. Kang, Duseok, DongHyun Kang, Jintaek Kang, Sungjoo Yoo, and Soonhoi Ha. "Joint optimization of speed, accuracy, and energy for embedded image recognition systems" In 2018 Design, Automation & Test in Europe Conference & Exhibition, 2018, ст. 715-720, (Препринт https://doi.org/10.23919/DATE.2018.8342102).
8. Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector" In European conference on computer vision, 2016, ст. 21-37, (Препринт https://doi.org/10.48550/arXiv.1512.02325).
9. Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection" In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, ст. 2117-2125, (Препринт https://doi.org/10.1109/ICCC51575.2020.9345302).

10. Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection" In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, ст. 779-788, (ПрепринтПрепринтhttps://doi.org/10.1109/CVPR.2016.91).
11. Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection" In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, ст. 10781-10790, (Препринт https://doi.org/10.1109/CVPR42600.2020.01079).
12. Abraham, Tara H. "(Physio) logical circuits: The intellectual origins of the McCulloch–Pitts neural networks" Journal of the History of the Behavioral Sciences 38, н. 1, 2002, (Препринт https://doi.org/10.1002/jhbs.1094).
13. Hebb, Donald Olding. "The organization of behavior: A neuropsychological theory" Psychology Press, 2005, (Препринт http://dx.doi.org/10.1186/s13041-020-00567-8).
14. Rasamoelina, Andrinandrasana David, Fouzia Adjailia, and Peter Sinčák. "A review of activation function for artificial neural network" In 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), ст. 281-286, 2020 (Препринт https://doi.org/10.1145/3492547.3492608).
15. Asadi, Behnam, and Hui Jiang. "On approximation capabilities of ReLU activation and softmax output layer in neural networks", 2022, (Препринт https://arxiv.org/abs/2002.04060).
16. Gu, Bonwoo, and Yunsick Sung. "Enhanced reinforcement learning method combining one-hot encoding-based vectors for cnn-based alternative high-level decisions" Applied Sciences 11, н. 3, 2021, (Препринт https://doi.org/10.3390/app11031291).
17. De Boer, Pieter-Tjerk, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. "A tutorial on the cross-entropy method" Annals of operations research 134, н. 1, 2005, (Препринт https://doi.org/10.1007/s10479-005-5724-z).
18. Beymer, David, Philip McLauchlan, Benjamin Coifman, and Jitendra Malik. "A real-time computer vision system for measuring traffic parameters" In Proceedings of IEEE computer society conference on computer vision and pattern recognition, 1997, ст. 495-501, (Препринт https://doi.org/10.1109/CVPR.1997.609371).
19. Andrychowicz, Marcin, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. "Learning to learn by gradient descent by gradient descent" Advances in neural information processing systems н. 29, 2016, (Препринт https://www.arxiv.org/abs/1606.04474).

20. Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions" International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems н. 6, 1997, г. 2: ст. 107-116, (Препринт http://dx.doi.org/10.1142/S0218488598000094).
21. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift" In International conference on machine learning, 2016, ст. 448-456, (Препринт https://arxiv.org/abs/1502.03167).
22. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation" In Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, ст. 3431-3440, (Препринт https://arxiv.org/abs/1411.4038).
23. Liu, Baoyuan, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. "Sparse convolutional neural networks" In Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, ст. 806-814, (Препринт https://arxiv.org/abs/1704.07724).
24. Torrey, Lisa, and Jude Shavlik. "Transfer learning" In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, 2010, ст. 242-264, (Препринт http://dx.doi.org/10.4018/978-1-60566-766-9.ch011).
25. Philipp, George, Dawn Song, and Jaime G. Carbonell. "The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions", 2017 (Препринт https://arxiv.org/abs/1712.05577).
26. Liu, Ziming & Li, Jinyang & Gao, Guangyu & Qin, Alex. "Action Recognition: You Only Look Once", 2019, (Препринт http://dx.doi.org/10.13140/RG.2.2.23477.83688/1).
27. Aurélien Géron "Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow" SBN 978-1-492-03264-9, Second Edition, Published by O'Reilly Media, Inc 2019; г. 14, ст. 489.
28. Ning, Chengcheng, Huajun Zhou, Yan Song, and Jinhui Tang. "Inception single shot multibox detector for object detection" In 2017 IEEE International Conference on Multimedia & Expo Workshops 2019 (ICMEW), 2019, ст. 549-554, (Препринт https://doi.org/10.1109/ICMEW.2017.8026312).
29. Du, Zexing, Jinyong Yin, and Jian Yang. "Expanding receptive field yolo for small object detection" In Journal of Physics: Conference Series, 2019, ст. 1314, номер 1, (Препринт https://doi.org/10.1088/1742-6596/1314/1/012202).
30. Kim, Seung-Wook, Hyong-Keun Kook, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. "Parallel feature pyramid network for object detection" In Proceedings of the European Conference on Computer Vision (ECCV), 2018, ст.

234-250, (Препринт https://openaccess.thecvf.com/content_ECCV_2018/papers/Seung-Wook_Kim_Parallel_Feature_Pyramid_ECCV_2018_paper.pdf).
31. Meng, Tianjian, Golnaz Ghiasi, Reza Mahjorian, Quoc V. Le, and Mingxing Tan. "Revisiting Multi-Scale Feature Fusion for Semantic Segmentation", 2022, (Препринт https://arxiv.org/abs/2203.12683).
32. Aber, James S., Irene Marzolff, and Johannes Ries. "Small-format aerial photography: Principles, techniques and geoscience applications" ISBN: 978-0-444-54260-2, Elsevier, 2010. г. 2 ст. 15.
33. FAIR1M: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery, ISPRS Journal of Photogrammetry and Remote Sensing, 2022, г. 164, ст. 116-130, (Препринт https://doi.org/10.1016/j.isprsjprs.2021.12.004).
34. Aurélien Géron "Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow" SBN 978-1-492-03264-9, Second Edition, Published by O'Reilly Media, Inc 2019; г. 14, ст. 465.
35. YOLO v3. implementation (MIT License) [Електроний ресурс] https://github.com/qqwweee/keras-yolo3 (переглянуто 01.04.2022).
36. Sentinel-5 is a European Earth observation mission [Електроний ресурс] https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-5/overview (переглянуто 12.04.2022).