ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Сезонно-специфічний сегмент багаторівневої архітектури нейронної мережі класифікації зображень місцевості»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології спеціальності
126 Інформаційні системи та технології ступеня вищої освіти магістр
групи 126ІСТмд_21
Шевченко В. Є.
Керівник: Слюсар В. І.
Рецензент: Муравльов В. В.

Полтава – 2023 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю скорочення трафіку та зменшення затримки в екосистемах інтернет речей (IoT). Зазвичай, для цього виконується інтеграція IoT штучного інтелекту (AI) та комп'ютерного зору (CV). З іншого боку, AI підвищує функціонал безпілотних платформ (UAV) і дозволяє ідентифікувати об'єкти на місцевості та покращити прийняття рішень у складних умовах довкілля. Однак, питання впливу на візуальні характеристики об'єктів сезонних змін, що можуть вносити помилки у роботу CV потребують додаткових досліджень. Все це свідчить про актуальність теми роботи.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в рамках науково-дослідної роботи «Управління стратегією інноваційного розвитку підприємств в контексті підвищення їх конкурентоспроможності на аграрному ринку, сталого розвитку та забезпечення продовольчої безпеки держави» (2021 р.), що фінансувалась господарськими договорами із замовниками, Концепції розвитку штучного інтелекту в Україні (розпорядження Кабінету Міністрів України № 1787-р від 29.12.2021), тематиці досліджень Навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей Кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є класифікація сезону зображень за допомогою нейронної мережі.

Завданнями кваліфікаційної роботи є:

– вибір архітектури нейронної мережі для класифікації сезону зображень;

– впровадження Transfer Learning;

– обґрунтування рекомендацій щодо реалізації сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості.

Об'єктом дослідження є процес класифікації сезону зображень.

Предметом дослідження є продуктивність нейронної мережі, що застосовується для класифікації сезону зображень.

Методами дослідження є аналітичний, інформаційно-пошуковий, методи синтезу та навчання нейронних мереж класифікації сезонних зображень, робота з хмарним сервісом Google Colaboratory.

Інформаційна база кваліфікаційної роботи сформована з ресурсів, що містять інформацію про згорткові нейронні мережі, інструментарій для виконання Transfer Learning.

Елементи наукової новизни роботи полягають у створенні моделей глибокого навчання згорткових нейронних мереж класифікації сезону зображень; структури сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості.

Практична значущість роботи полягає у розробці рекомендацій щодо практичної реалізації проєкту Edge Computing + AI CV, що містить зазначений сезонно-специфічний сегмент і можуть бути використані для подальших досліджень за даною тематикою та при проектуванні хмарних сервісів.

Апробація результатів відбувалася в рамках V-ої Міжнародної студентської конференції «Теоретичне та практичне застосування результатів сучасної науки» (жовтень 2023 р., м. Рівне), І Міжнародної науково-практичної конференції «Стратегічний менеджмент агропродовольчої сфери в умовах глобалізації економіки: безпека, інновації, лідерство» (вересень 2023 р., м. Полтава).

За результатами досліджень здійснено 2 публікації тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 71 сторінку формату А4. Вона містить 42 рисунки.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ІНТЕГРАЦІЇ ШТУЧНОГО ІНТЕЛЕКТУ ТА КОМП'ЮТЕРНОГО ЗОРУ

1.1 Особливості застосування Edge Computing в проєктах інтернету речей

У сучасному світі, де кількість даних, що генеруються пристроями інтернет речей (IoT) [1], збільшується експоненційно, виникає критична необхідність обробки цих даних ближче до джерела, щоб зменшити затримку і скоротити трафік, що передається в хмару [2]. Це і є суть Edge Computing – обчислювальні процеси відбуваються на «краю» мережі, в безпосередній близькості до пристроїв, що генерують дані. З розвитком мікропроцесорних технологій деякі пристрої на краю мережі тепер здатні виконувати більш складні обчислення. На даний час, спостерігається інтеграція IoT з іншими галузями, наприклад: штучним інтелектом (AI) [3] і комп'ютерним зором (CV) [4]. Наприклад, промислові ПК або спеціалізовані Edge-сервери можуть використовувати AI для більш точного аналізу зображень у реальному часі [5]. Це створює потужні системи, здатні збирати, аналізувати та відповідати на візуальні дані, реалізувати інтелектуальне управління енергоресурсами значно розширюючи можливості та ефективність IoT-рішень. Це дозволяє виявляти тенденції, прогнозувати поведінку системи та оптимізувати процеси. AI може автоматизувати взаємодію між пристроями IoT, полегшуючи управління ними та дозволяючи їм самонавчатися на основі зібраних даних. AI може допомогти в оптимізації роботи IoT систем, знижуючи витрати на енергію та ресурси та підвищуючи загальну продуктивність, AI дозволяє розробляти нові рішення та сервіси на основі даних з IoT пристроїв, пропонуючи інноваційні способи використання цих технологій, AI може посилити безпеку IoT систем, аналізуючи дані на предмет виявлення потенційних загроз та автоматично реагуючи на них.

Проекти, що поєднують Edge Computing + AI + CV продовжують набирати популярності в різних сферах, що дає низку переваг [6].

1. Приватність та безпека. Обробка даних на пристрої дозволяє уникнути передачі чутливих даних у хмару, що зменшує ризики, пов'язані з конфіденційністю та безпекою даних. У системах безпеки, де потрібна висока точність у розпізнаванні осіб або інших біометричних даних, AI може використовуватися для підвищення точності та надійності систем на краю мережі. Для систем відеоспостереження, оснащених можливостями розпізнавання об'єктів або осіб, AI дозволяє швидко та точно аналізувати відеопотік на пристрої, мінімізуючи затримки та потребу передачі даних на віддалений сервер.

2. Зменшення затримки. Розміщення алгоритмів CV безпосередньо на краю мережі дозволяє обробляти дані майже в реальному часі, що є критично важливим для додатків, що потребують миттєвої реакції, наприклад, в автономному водінні або робототехніці. Відстеження запасів, управління чергами та персоналізований маркетинг – все це сфери, де Edge Computing з використанням AI + CV, може надати значні переваги, опрацьовуючи дані безпосередньо на місці.

3. Зниження навантаження на мережу. Оскільки дані обробляються локально, не потрібно постійно передавати велику кількість даних до центрального сервера, що знижує навантаження на мережну інфраструктуру У контексті розумних міст, де камери та сенсори поширені повсюдно, використання Edge Computing з AI + CV може допомогти в реалізації функцій, таких як аналіз трафіку, керування паркуванням та моніторинг безпеки в режимі реального часу.

4. Економія ресурсів. Edge Computing дозволяє заощадити ресурси центральних хмарних серверів, оскільки обробка відбувається безпосередньо на периферійних пристроях.

5. Широкий спектр додатків. Застосування AI CV на краю мережі може бути знайдено в таких індустріях, як виробництво (наприклад,

дефектоскопії), роздрібна торгівля (автоматизація через системи розпізнавання товарів), розумні міста (системи відеоспостереження з розпізнаванням осіб та об'єктів), охорона здоров'я, та багатьох інших [7]. У медичних пристроях, які повинні працювати в реальному часі та часто діють автономно, AI може служити для аналізу зображень, наприклад, для діагностики або моніторингу пацієнтів. У медичних додатках, де надзвичайно важливою є точність аналізу зображень (наприклад, при скануванні МРТ або КТ), AI може використовуватися для попередньої обробки та аналізу даних на пристроях, що знаходяться безпосередньо в лікарнях або діагностичних центрах.

6. Розвиток IoT. Завдяки розвитку IoT все більше пристроїв здатні виконувати обчислення на краю мережі, що збільшує можливості інтеграції AI та CV у різні пристрої та системи. Для управління якістю та оптимізації процесів на виробництві може забезпечити швидку та ефективну обробку зображень безпосередньо на виробничій лінії. Як частина системи CV на виробництві, AI може допомогти у складному завданні дефектоскопії, де потрібна висока точність для ідентифікації дрібних дефектів на деталях або складальних одиницях.

8. Стійкість до збоїв. Розподілена природа Edge Computing робить системи менш уразливими до збоїв центральних серверів та мережних перебоїв.

9. Підтримка в реальному часі. У багатьох випадках (наприклад, у промисловості або моніторингу безпеки), потрібен миттєвий зворотний зв'язок, який може забезпечити система на базі Edge Computing та AI + CV.

7. Прогрес в апаратному забезпеченні. Поява спеціалізованих чіпів та апаратного забезпечення для обробки AI алгоритмів на краю мережі (наприклад, TPU від Google, VPU від Intel) забезпечує більш високу ефективність та продуктивність.

В області безекіпажних та безпілотних систем (UAV) AI може застосовуватися для розширеного розпізнавання зображень та обробки

відеопотоку, щоб точно ідентифікувати об'єкти на дорозі та покращити прийняття рішень у складних умовах довкілля.

1.2 Вибір архітектури для завдань Object Detection

Ultralytics нещодавно випустила сімейство моделей виявлення об'єктів YOLOV8 [8]. Ці моделі перевершують попередні версії моделей YOLO як за швидкістю, так і за точністю в наборі даних COCO [9]. Я к відомо, у галузі CV та Object Detection [10], що швидко розвивається, алгоритм You Look Only Once (YOLO) змінив правила гри. YOLO, відомий своїми можливостями виявлення об’єктів у реальному часі, протягом багатьох років пережив кілька ітерацій, кожна з яких розсувала межі точності та швидкості. Серед цих ітерацій YOLO v8 постає як останнє диво, демонструючи невпинне прагнення до досконалості у виявленні об’єктів на основі глибокого навчання. Алгоритм YOLO, представлений у 2016 р., зробив революцію у виявленні об’єктів, запропонувавши уніфіковану модель, здатну передбачати обмежувальні прямокутники та ймовірності класу безпосередньо з вхідного зображення в режимі реального часу. Ця зміна парадигми, відхід від регіональних методів до однопрохідних, наскрізних нейронних мереж, принесла значні переваги з точки зору швидкості та ефективності. YOLO v8 базується на основній концепції YOLO, розробленій Ultralytics, додатково вдосконалюючи її для досягнення оптимальної продуктивності. По суті, YOLO v8 працює, розділяючи вхідне зображення на сітку, як правило, розміром 13×13 або 26×26, залежно від конкретного типу. Кожна комірка сітки бере на себе відповідальність за прогнозування об’єктів у своїй просторовій області.

1. Обробка вхідних даних. YOLO v8 приймає зображення як вхідні дані та поділяє його на сітку, як правило, розміром 13×13 або 26×26, залежно від конкретного типу. Кожна комірка сітки відповідає за прогнозування об’єктів у своїй просторовій області.

2. Виділення функцій. Мережа витягує високорівневі характеристики з вхідного зображення за допомогою глибокої згорткової нейронної мережі (CNN) [11]. Вибір архітектури мережі часто береться з усталених моделей, таких як ResNeXt [12] або Darknet [13].

3. Прогнозування обмежувальної рамки. YOLO v8 прогнозує обмежувальну рамку для об'єктів шляхом регресії координат верхнього лівого кута, ширини та висоти рамки. Крім того, він обчислює оцінку достовірності, яка вказує на ймовірність того, що передбачений ящик містить об'єкт.

4. Прогнозування класу. Поряд із прогнозуванням обмежувальної рамки YOLO v8 також прогнозує ймовірності класу для кожної комірки сітки. Це означає, що модель може не тільки виявляти об'єкти, але й ідентифікувати їх відносні категорії.

5. Постобробка. Після того, як прогнози зроблені, застосовується поріг достовірності, щоб відфільтрувати виявлення з низькою достовірністю. Немаксимальне придушення потім використовується для видалення повторюваних або накладених обмежувальних рамок, забезпечуючи продовження лише найточнішого пошуку.

На даний час, існує кілька варіантів YOLO v8, кожен з яких ретельно розроблений відповідно до конкретних випадків використання та потреб.

1. YOLO v8-Tiny робить компроміс між точністю та швидкістю передбачення. Це досягається завдяки застосуванню невеликої мережевої архітектури та малого розміру сітки (наприклад, 13×13), що забезпечує продуктивність у реальному часі навіть на пристроях з обмеженими ресурсами.

2. YOLO v8-SPP. Тип просторового об'єднання пірамід (SPP) YOLO v8 включає модуль SPP у мережі. Це доповнення полегшує ефективне захоплення багатомасштабних функцій, що сприяє підвищенню точності, особливо для об'єктів різного розміру.

3. YOLO v8-CSPDarknet. YOLO v8-CSPDarknet – це поєднання архітектури YOLO v8 із магістраллю CSPDarknet. Це злиття забезпечує чудову продуктивність і точність. Включення CSP (Cross-Stage Partial) підключення покращує функцію представлення, що робить його ідеальним вибором для різноманітних застосувань.

4. YOLO v8-Panet. YOLO v8-Panet інтегрує архітектуру PANET (мережі агрегації шляхів), яка підвищує точність об'єднання функцій і виявлення об'єктів. Ця версія найкраща в сценаріях, де точна локалізація об'єктів має першочергове значення.

YOLO v8 має переваги перед попередніми версіями. Це являє собою значний крок вперед у технології виявлення об'єктів порівняно з попередником. Ось деякі з основних переваг.

1. Підвищена точність. Варіанти YOLO v8, такі як v8-CSPDarknet і v8-Panet, пропонують підвищену точність завдяки вдосконаленому представленню функцій і методам об'єднання. Ця висока точність робить YOLO v8 більш надійним для складних застосувань, таких як автономні транспортні засоби та медична візуалізація.

2. Продуктивність у режимі реального часу. YOLO v8 зберігає відмітну продуктивність YOLO в режимі реального часу, досягаючи кращої точності. Ця характеристика робить його ідеальним для реальних додатків, які потребують як швидкого, так і точного виявлення об'єктів, таких як системи спостереження та робототехніка.

3. Універсальність. Завдяки ряду варіантів, що відповідають конкретним потребам, YOLO v8 можна налаштувати відповідно до різноманітних програм. Незалежно від того, чи потрібен вам легкий YOLO v8-Tiny для вбудованих пристроїв чи високоточний YOLO v8-CSPDarknet для досліджень і розробок, є варіант, який задовольнить потреби.

4. Покращена надійність. Інтеграція розширених функцій, таких як підключення CSP і архітектура Panet у варіантах YOLO v8, робить модель

більш надійною проти перешкод, захаращених сцен і складних умов освітлення.

Команда Ultralytics порівняла YOLOv8 із базою даних COCO та досягла вражаючих результатів порівняно з попередніми версіями YOLO для всіх 5-ти розмірів моделей (рис. 1.1, 1.2) [14].

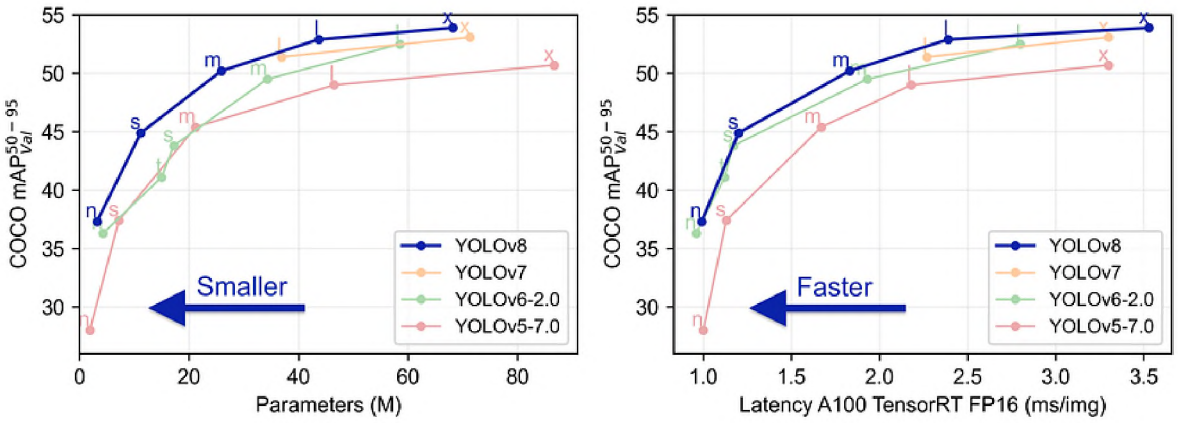

Рисунок 1.1 – Порівняння версій YOLO

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

- **mAP^{val}** values are for single-model single-scale on COCO val2017 dataset.
  Reproduce by `yolo val detect data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an Amazon EC2 P4d instance.
  Reproduce by `yolo val detect data=coco128.yaml batch=1 device=0/cpu`

Рисунок 1.2 – Показники продуктивності різних поколінь YOLO

Коли порівнюють продуктивність різних поколінь YOLO та розмірів моделей на датасеті COCO, зазвичай, використовують три показники:

– продуктивність – середня точність (mAP);

– швидкість –швидкість висновку (у кадрах/с);

– обчислення (вартість) – розмір моделі у FLOP і параметрах.

При порівнянні виявлення об’єктів модель YOLOv8m досягла mAP = 50,2% на наборі даних COCO, тоді як найбільша модель, YOLOv8x досягла 53,9% із більш ніж удвічі більшою кількістю параметрів (див. рис. 1.2). Загалом висока точність і продуктивність YOLOv8 роблять її сильним претендентом на проєкт AI + CV. Оскільки CV продовжує розвиватися, YOLO v8 обіцяє бути в авангарді, розсуваючи межі того, що можливо у виявленні та розпізнаванні об’єктів. Його принцип роботи та діапазон варіацій роблять його маяком інновацій у галузі, освітлюючи шлях до більш точних, ефективних і універсальних рішень для виявлення об’єктів. Щоб отримати додаткову інформацію, доступно багато ресурсів.

1.3 Обґрунтування реалізації сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі для завдань класифікації зображень місцевості

Сезонні зміни суттєво впливають на візуальні характеристики об'єктів, що можуть вносити помилки у роботу систем Object Detection (рис. 1.3).

Рисунок 1.3 – Наявність туману

Сонячне світло та його інтенсивність змінюються протягом року, впливаючи на освітленість сцени. Зимові місяці можуть характеризуватися коротким днем та низьким кутом падіння сонячних променів, що створює довгі тіні та змінює контрасти на зображенні (рис. 1.4). Літні місяці мають більш яскраве освітлення, що може призвести до пересвітлених ділянок. Такі умови освітлення можуть впливати на точність визначення кольору та форми об'єктів.

Дощ, туман, снігопад – усі ці погодні умови можуть змінюватися сезонно та впливати на видимість об'єктів. Наприклад, туман може значно знижувати контрастність та розрізнення об'єктів, в той час як снігопад може покривати та деформувати візуальні ознаки, за якими алгоритм розпізнає об'єкти.

Рисунок 1.4 – Сніг на поверхні ґрунту

Сезонні зміни в рослинності та покритті землі можуть маскувати або відкривати нові об'єкти в полі зору камери. Літнє зелене листя та густа трава можуть приховувати дорожні знаки або інші об'єкти, тоді як осіннє опадання листя відкриває раніше закриті види.

Сезонні зміни впливають на кольори та текстури у навколишньому середовищі, що може впливати на здатність алгоритмів виявлення об'єктів відрізняти об'єкти від фону.

Для того, щоб системи Object Detection могли надійно працювати в усіх сезонах, необхідно зробити так, щоб вони могли адаптуватися до цих змін. Створення сезонного набору даних є одним із ключових етапів у підвищенні ефективності систем Object Detection. Цей процес включає кілька кроків. Розробка методології збору та анотації сезонних зображень для створення представницького набору даних. Перед початком збору даних необхідно визначити, які сезони і погодні умови представлятимуть найбільший інтерес для подальшого аналізу і використання. Наприклад, для аграрного сектору важливими будуть зображення посівів у різні періоди їх росту та збору врожаю, в той час як для міської інфраструктури може бути корисним аналіз стану доріг у зимовий період. Збір зображень може відбуватися через різноманітні джерела: супутникові знімки, аерофотознімання, знімки з беспілотних літальних апаратів або наземні фотографії. Для забезпечення різноманітності набору даних, зображення повинні бути отримані в різні часові періоди і за різних погодних умов. Кожне зібране зображення необхідно анотувати, що включає в себе відмітку об'єктів, які повинні бути розпізнані системою Object Detection. Анотація може бути ручною або напівавтоматичною з використанням попередньо навчених моделей для підвищення ефективності процесу. Після анотації проводиться перевірка на помилки і неточності. Цей крок важливий, оскільки від якості анотації безпосередньо залежить якість навчання нейронної мережі. Для підвищення узагальнюючої спроможності моделей розпізнавання об'єктів використовують розширення даних. Використання розширених технік передобробки та аугментації зображень для підвищення їх різноманітності та якості. Це може включати зміну масштабу зображень, обертання, зміни освітлення та колірної гами, що допомагає моделі краще адаптуватися до різноманітних умов. Перед використанням набору даних у навчанні моделей виконується його валідація.

Вона може включати аналіз розподілу класів, перевірку на збалансованість даних та виявлення можливих викидів.

1.4 Вибір архітектури для класифікатора сезону

Визначення топології мережі орієнтується на вирішуване завдання, дані з наукових статей та власний експериментальний досвід. Можна виділити такі етапи що впливають вибір топології:

– визначити задачу, що вирішується нейромережею (класифікація, прогнозування, модифікація);

– визначити обмеження у розв'язуваній задачі (швидкість, точність відповіді);

– визначити вхідні (тип: зображення, звук, розмір: 100×100, 30×30, формат: RGB, в градаціях сірого) та вихідні дані (кількість класів).

Існує величезна різноманітність різних архітектур (рис. 1.5).

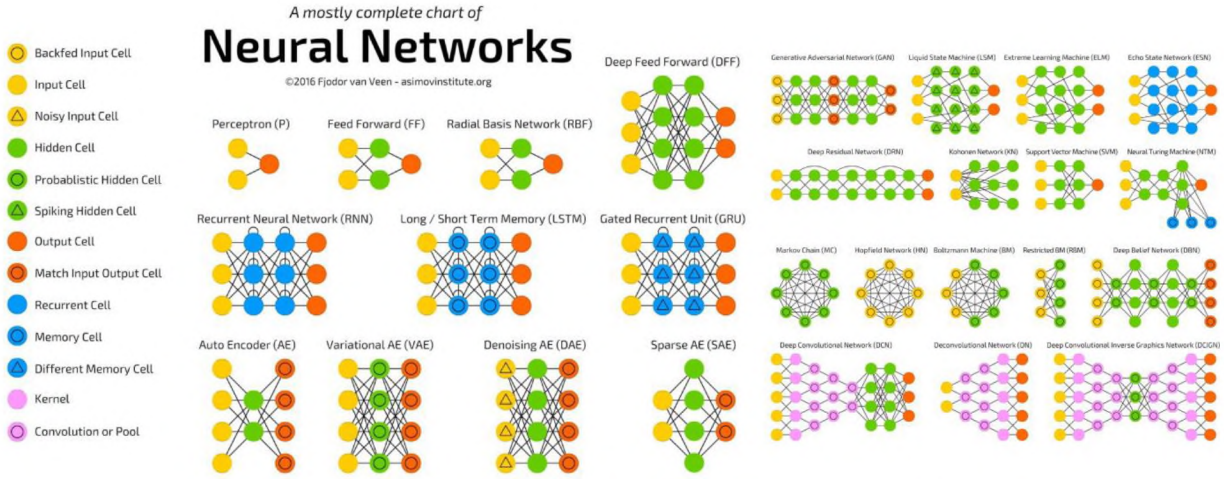

Рисунок 1.5 – Архітектури нейронних мереж

Основним призначенням Convolutional Neural Network (CNN) є виділення вихідного зображення малих частин, що містять опорні (характерні) ознаки (features), такі як ребра, контури, дуги або грані. На наступних рівнях обробки з цих ребер можна розпізнати складніші

повторювані фрагменти текстур (кола, квадратні фігури та ін.), які далі можуть скластися ще складніші текстури (частина особи, колесо машини та ін.). Приклад реалізації класифікатору на основі ансамблю згорток наведено на рис. 1.6.

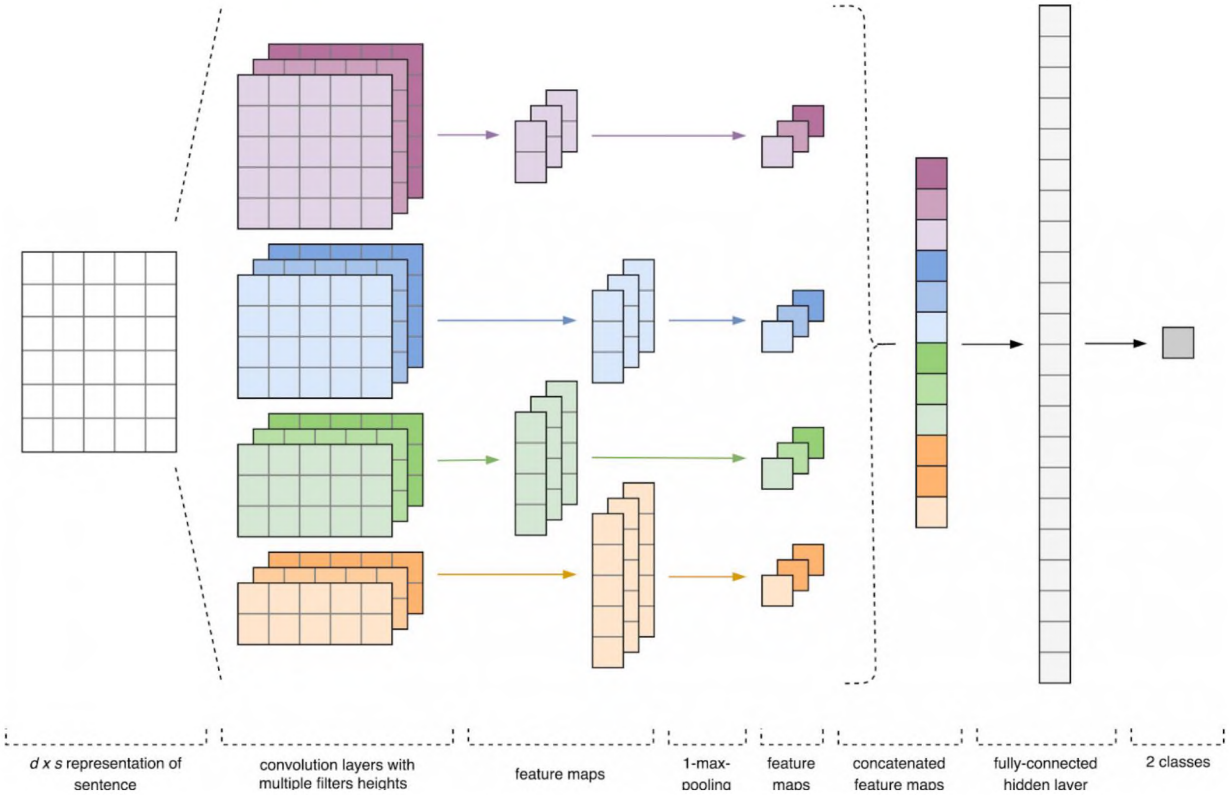

Рисунок 1.6 – Варіант архітектури CNN

Причиною успіху CNN стала концепція загальної ваги. Незважаючи на великий розмір, ці мережі мають невелику кількість параметрів, що налаштовуються. CNN можуть швидко працювати на послідовній машині і швидко навчатися за рахунок чистого розпаралелювання процесу згортки по кожній карті ознак, а також зворотного згортки при поширенні помилки по мережі. Базовим елементом CNN є фільтр (матричної форми), який просувається вздовж зображення с кроком в один піксел (комірку) вздовж горизонтальної та вертикальної осях, починаючи від верхнього лівого кута і закінчуючи нижнім правим. На кожному кроці – CNN виконує обчислення з метою оцінки – на скільки схожа частина зображення фільтра, що потрапили у вікно, з патерном самого фільтра.

Підвибірковий шар також, як і згортковий карти, але їх кількість збігається з попереднім (згортковим) шаром, їх 6. Мета шару - зменшення розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібне і воно ущільняється до менш докладного. До того ж, фільтрація вже непотрібних деталей допомагає не перевчитися. У процесі сканування ядром підвибіркового шару (фільтром) карти попереднього шару, скануюче ядро не перетинається на відміну від згорткового шару. Зазвичай кожна карта має ядро розміром 2×2, що дозволяє зменшити попередні карти згорткового шару в 2 рази. Вся карта ознак поділяється на комірки 2×2 елементи, у тому числі вибираються максимальні за значенням. Зазвичай, у підвибірковому шарі застосовується функція активації ReLU. Вихідний шар пов'язаний із усіма нейронами попереднього шару. Кількість нейронів відповідає кількості класів, що розпізнаються, тобто 2 – сніг і не сніг. Але для зменшення кількості зв'язків та обчислень для бінарного випадку можна використовувати один нейрон і при використанні як функцію активації гіперболічний тангенс, вихід нейрона зі значенням -1 означає приналежність до класу «не сніг», а вихід нейрона зі значенням 1 – означає приналежність до класу осіб.

Одним із етапів розробки нейронної мережі є вибір функції активації нейронів. Вигляд функції активації багато в чому визначає функціональні можливості нейронної мережі та метод навчання цієї мережі. Класичний алгоритм зворотного поширення помилки добре працює на двошарових та тришарових нейронних мережах, але при подальшому збільшенні глибини починає відчувати проблеми. Однією з причин є так зване згасання градієнтів. У міру поширення помилки від вихідного шару до вхідного на кожному шарі відбувається збільшення поточного результату на похідну функції активації. Похідна у традиційної сигмоїдної функції активації менше одиниці по всій області визначення, тому після кількох шарів помилка стане

близькою до нуля. Якщо ж, навпаки, функція активації має необмежену похідну (як, наприклад, гіперболічний тангенс), може статися вибухове збільшення помилки у міру поширення, що призведе до нестійкості процедури навчання.

CNN має різні архітектури, які відіграли ключову роль у побудові алгоритмів, на яких стоїть і в найближчому майбутньому стоятиме AI, наприклад: LeNet [15], AlexNet [16], VGGNet [17], GoogLeNet [18], ResNet [19], ZFNet [20] та ін. Однак, не кожна архітектура CNN може бути використана в проєктах Edge Computing + AI + CV.

Також доцільно звернути увагу на Transfer Learning (перенесення навчання), яка дозволяє адаптувати вже попередньо навчені моделі до нових сезонних умов, значно зменшуючи потребу в обсязі даних та обчислювальних ресурсах для повторного навчання. Це потужний метод у галузі глибокого навчання, особливо корисний для завдань, пов'язаних із CNN. В даному контексті слід звернути увагу на кілька аспектів.

1. Використання попередньо вивчених моделей. Трансферне навчання часто включає використання моделі, попередньо вивченої на великому та різноманітному наборі даних, наприклад, ImageNet [21]. Ці моделі, такі як VGG, ResNet або Inception [22], вже навчилися отримувати корисні ознаки з зображень і можуть бути гарною відправною точкою для нових завдань.

2. Модифікація та налаштування. Для конкретної задачі ви можете змінити останні верстви попередньо навченої моделі, щоб вони краще відповідали вашим специфічним вимогам. Наприклад, ви можете замінити вихідний шар, щоб він відповідав кількості класів у вашій задачі класифікації.

3. Навчання із замороженими шарами. На початку ви можете «заморозити» ваги більшої частини мережі, навчаючи лише останні шари. Це допомагає уникнути забування вже вивчених ознак та прискорює процес навчання.

4. Тонка настройка. Після початкового тренування останніх шарів можна «розморозити» частину або всі шари мережі, що залишилися, і продовжити навчання на вашому наборі даних. Це дозволяє мережі більш точно адаптуватися до специфіки вашого завдання.

5. Дані та регуляризація. Для ефективного трансферного навчання важливо використовувати достатній та релевантний набір даних. Також рекомендується застосовувати методи регуляризації, такі як dropout або регуляризація ваги, щоб уникнути перенавчання.

6. Експериментування та ітерації. Налаштування процесу трансферного навчання може вимагати багаторазових експериментів з різними архітектурами, гіперпараметрами та стратегіями навчання, щоб знайти найкраще рішення для вашого конкретного завдання.

Трансферне навчання особливо корисне в ситуаціях, коли є обмежена кількість навчальних даних, або коли навчання мережі з нуля вимагає значних обчислювальних ресурсів. Воно дозволяє значно прискорити процес навчання та покращити точність моделей CNN на специфічних завданнях.

Transfer Learning є методом, що дозволяє передавати знання з однієї області (домену) до іншої. У контексті нашої задачі, ми використовуємо моделі глибокого навчання мережі, які були попередньо навчені на великих та різноманітних наборах даних (наприклад, ImageNet), для ініціалізації вагів у нових сезонних наборах даних. Цей підхід значно покращує здатність моделі адаптуватись до сезонних змін, не «забуваючи» при цьому первинно навчені особливості. Для адаптації навчених моделей до сезонних умов ми використовуємо тонке налаштування (fine-tuning), при якому верхні шари моделі тренуються заново з новим сезонним набором даних. Це дозволяє моделі деталізувати свої знання, адаптуючи їх до специфічних особливостей кожного сезону, таких як засніжені пейзажі, листя, особливості освітлення та ін. Тобто сезонні датасети, які містять велику кількість анотованих зображень з різних сезонів.

Таке розширення наборів даних є критичним для навчання нейронних мереж, щоб вони могли ефективно розрізняти об'єкти в змінних умовах. Після адаптації моделі за допомогою Transfer Learning [23] проводимо ретельне оцінювання її продуктивності на валідаційних і тестових сезонних наборах даних.

Висновки до розділу 1

Для скорочення трафіку в системах IoT застосовується Edge Computing. Інтеграція IoT з AI і CV дає низку переваг. В UAV AI може застосовуватися для розширеного розпізнавання зображень та обробки відеопотоку, щоб точно ідентифікувати об'єкти на дорозі та покращити прийняття рішень у складних умовах довкілля.

Висока точність і продуктивність YOLOv8 роблять її сильним претендентом на проєкт AI + CV при вирішенні завдань Object Detection і сегментації.

Сезонні зміни суттєво впливають на візуальні характеристики об'єктів, що можуть вносити помилки у роботу систем Object Detection. Для побудови класифікатора сезону проведено вибір архітектури нейронної мережі. В якості базової розглядається CNN. Однак не кожна архітектура CNN може бути використана в проєктах Edge Computing + AI + CV. Це вимагає проведення додаткових досліджень.

Для прискорення процесу навчання та покращення точність моделей CNN на завданнях Object Detection і сегментації доцільно використовувати Transfer Learning.

РОЗДІЛ 2
СИНТЕЗ СЕЗОННО-СПЕЦИФІЧНОГО СЕГМЕНТУ БАГАТОРІВНЕВОЇ АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ МІСЦЕВОСТІ

2.1 Деталізація структури сезонно-специфічного сегменту

На основі проведених досліджень в роботі визначено структуру сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості. Вона містить два блоки. Перший блок проводить класифікацію сезону, що відповідає зображенню. В залежності від вибору, у другому блоці активується відповідна до сезону модель нейронної мережі за основним заданням, наприклад, реалізація автопілоту, Object Detection, Object Tracking [24] та ін. Такий підхід дозволяє знизити вимоги до обчислювальних потужностей Edge Computing, зберігаючи продуктивність проєктів Edge Computing + AI CV. При цьому, необхідно вибрати архітектуру мережі для реалізації першого блоку.

В даному випадку, для виконання завдання класифікації сезону доцільно розглянути згорткові нейронні мережі (CNN). Це пов'язано з кількома наступними чинниками.

1. Використання локальних особливостей. CNN здатні виявляти локальні особливості на зображеннях, такі як форми, текстури, кольори та візерунки. Це особливо корисно для розпізнавання сезонних характеристик, наприклад, опалого листя восени, снігу взимку, цвітіння квітів навесні або зелених листя влітку.

2. Збереження просторових відносин. Згорткові шари в CNN зберігають просторові відносини між пікселями, що дозволяє мережі визначати об'єкти і їх контекст. Це значно полегшує розрізнення сезонних особливостей, які часто залежать від просторової конфігурації елементів сцени.

3. Шари активації. Функції активації, такі як ReLU [25], допомагають мережі вчитися на нелінійних даних. Це особливо корисно для складних зображень, де сезонні особливості можуть варіюватися за формою, освітленням і кольором.

4. Зменшення розмірності. CNN автоматично зменшують розмірність даних через пулінгові шари, що допомагає у вилученні більш абстрактних особливостей із зображення та зменшує ризик перенавчання.

5. Здатність до генералізації. Після навчання на різноманітному наборі даних, CNN здатні ефективно генералізувати та правильно класифікувати нові, невідомі зображення. Це робить їх ідеальними для класифікації сезону, навіть якщо зображення має унікальні особливості або умови освітлення.

Ці характеристики роблять CNN особливо потужними для задач, пов'язаних з обробкою зображень, включаючи класифікацію сезонів.

2.2 Архітектура MobileNetV2

У свій час, MobileNet [26] породив революцію, ставши прикладом швидкої та ефективної нейронної мережі. Щоб зручно запускатися на мобільних пристроях, сетка повинна влезти в пам'ять і її інференс повинен брати яке-то розумне час. Щоб є, потрібно щоб у мережі було менше параметрів і матричних умножень. При цьому хочеться отримати максимальну якість.

У 2017 р. дослідники з Google зробили перший вагомий крок у цей бік – випустили MobileNetV1 [27]. У мережі активно використовувалися глибинні роздільні згортки, про які ми розповідали в цьому посту. На рис. 2.1 видно побудову блока мережі. Варто зауважити, що між deepwise conv і 1×1 conv є нелінійність – ця дільниця значно перероблена в наступній версії.

Цікаво, що автори звертали увагу не тільки на кількість, але і на реалізацію параметрів зворотного шару. Згортки представлені одним

матричним добутком. Але щоб перетворити згортку в матричний добуток потрібно перегрупувати ядро так, щоб при перемноженні зробити симуляцію переміщення ядра по вхідному тензору. Детальніше про те, як це роблять.

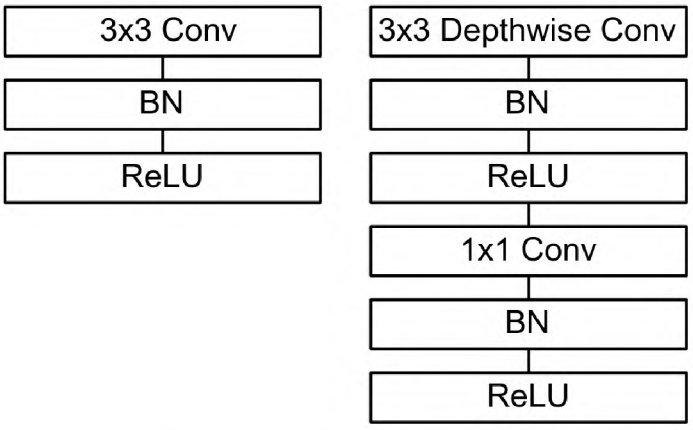

Рисунок 2.1 – Стандартний блок звичайної мережі (зліва) і MobileNet (справа)

Ця імітація вимагає дублювання деяких значень тензора, що збільшує потребу пам'яті та додає понад кількість операцій. Но згорткові шари з ядром 1×1 відрізняються тим, що дозволяють вважати згортку як матричний добуток без перегрупування даних. Завдяки тому, що 75 % параметрів MobileNet відводяться під згортки 1×1, набір працює дуже швидко та без обмеження пам’яті. Крім того, вже в першій статті є ідеї підбору розміру мережі під задачу. Автори вводять два параметра, відповідальних за розмір шарів. Обидва лежать у діапазоні [0, 1], де одиниця відповідає стандартному MobileNetV1. Перший з них, Width Multiplier, повинен рівномірно стягнути згортку, зменшивши кількість каналів на кожному шарі.

Далі зменшуємо кількість згорток 1×1 у Width Multiplier раз, із-за чого кількість вхідних і вихідних каналів зменшується лише на одне. По-друге, множник роздільної здатності відповідає просто за вхідний розмір зображення, його зменшення рівноцінно ресайзу до розміру 224 × множник роздільної здатності. MobileNet облаштував багато ще використовуваних на цей момент архітектур з точністю, при цьому ім'я викликає мало операцій на

добуток та параметрів мережі. Менше ніж через рік автори першої статті вирішили закріпити успіх і додали поверх наявної архітектури ще пару властивостей, які відображені у назві – «MobileNetV2: інвертовані залишки та лінійні вузькі місця» [29]. Основна ідея лежить в тому, що нелінійність знищує інформацію в каналі. Наприклад, отримавши тільки після застосування ReLU, вже не можемо сказати, яке негативне число було до функцій активації. Тоді можна попробувати кілька підходів.

1. Збільшити розмір внутрішнього представлення, щоб при «очищені» інформації в одному каналі, яка зберігалася в іншому.

2. Використовуйте лінійні шари, щоб «зібрати» корисну інформацію з тензора після ReLU. Ведь тепер у ньому багато нулів, які не можуть нести корисну інформацію.

На рис. 2.2 навадена реалізація цих ідей. На відміну від звичайного блоку ми застосовуємо нелінійність тільки до тензорів з великим числом каналів, для інших залишаємо на лінійному перетворенні. Крім того, перед застосуванням активації ми використовуємо згортки 1x1 без нелінійності для збільшення числа каналів.

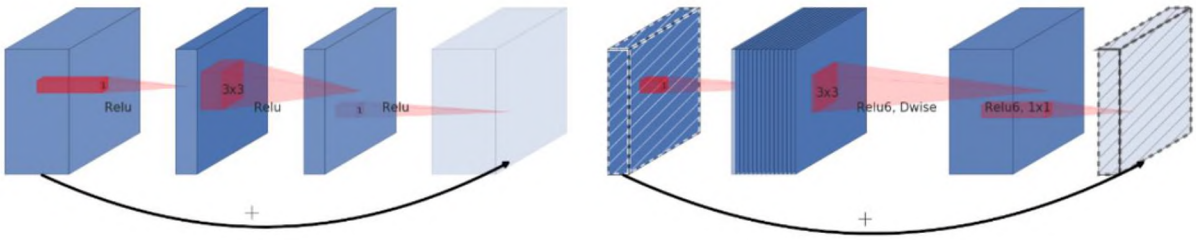

Рисунок 2.2 – Residual block: в інших мережах і в MobileNetV2

Коли зменшуємо розмір внутрішнього представлення мережі в MobileNetV1 за допомогою множника ширини, ми сподіваємося, що вся ціна для мережевої інформації може бути поміщена в просторово низькі розміри. Так само і в новій версії — сполучна через залишкове з'єднання саме «тонкі» тензори, ми сподіваємося, що вся ціна інформація буде зберігатися в них. Це поєднується з ідеєю, що нелінійність цієї інформації може викреслити – адже як раз до цих блоків вона не застосовується (рис. 2.3). Крім успіхів у

класифікації на ImageNet, автори показали результати в виявленні і семантичних сегментах. Як видно з рис. 2.4, їх версія SSDLite не тільки обігнала YOLOv2 по mAP, але і залишилася в кілька разів менше і швидше. Це стало зручним переносом на мобільні пристрої не тільки завдань класифікації, але й більш складних.

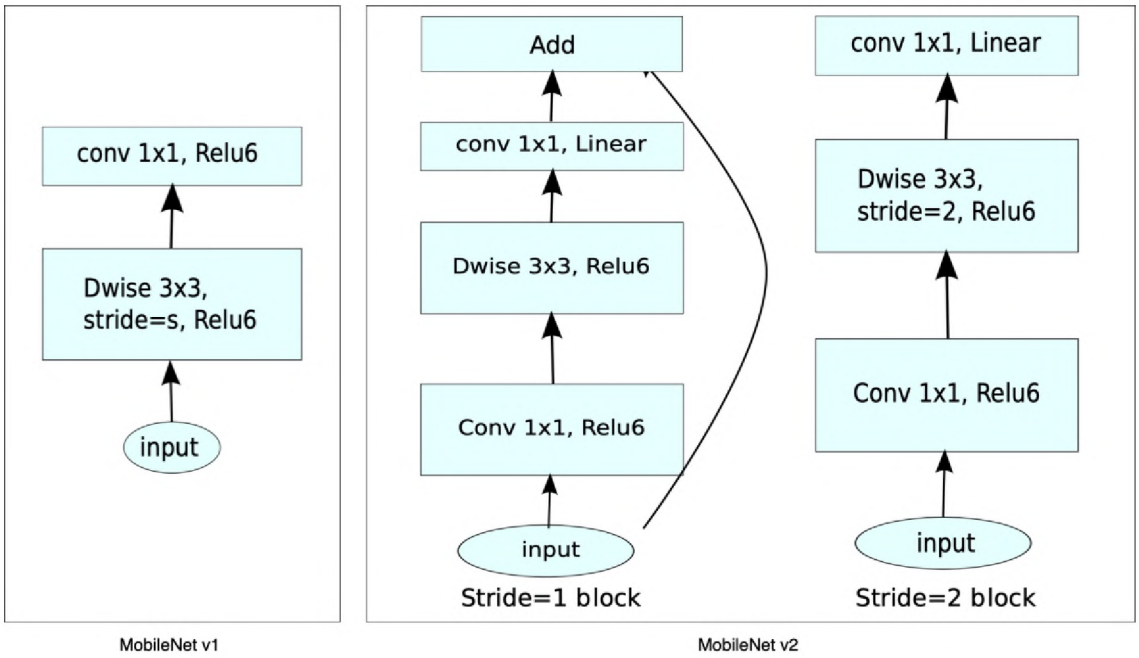

Рисунок 2.3 – Блок MobileNet і варіанти блоку з різними страйдом в MobileNetV2 (справа)

Network	mAP	Params	MAdd	CPU
SSD300	23.2	36.1M	35.2B	-
SSD512	26.8	36.1M	99.5B	-
YOLOv2	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Рисунок 2.4 – Задача детекції в реальному часі на COCO («MNet +» обозначає використання MobileNet в якості магістралі)

MobileNetV2 є прикладом CNN, яка була спроектована для використання в мобільних та вбудовуваних пристроях. Вона була представлена у статті [29] і розроблена в Google. Вона є прямим

продовженням MobileNetV1 та була представлена у 2018 р. MobileNetV2 – це ефективна архітектура згорткової нейронної мережі, розроблена командою дослідників Google для використання в мобільних та вбудовуваних пристроях. MobileNetV2 є покращенням оригінальної архітектури MobileNet і вводить такі концепції, як інвертовані залишкові зв'язки та лінійні вузькі місця (рис. 2.5).

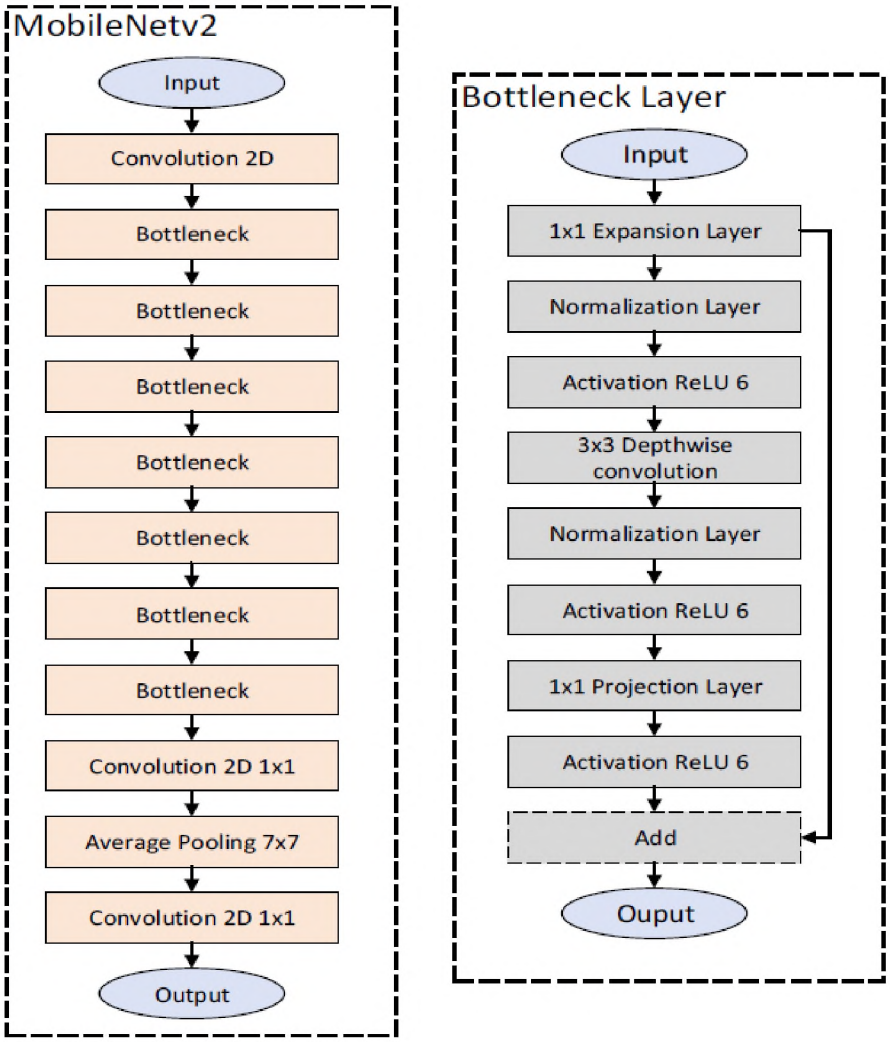

Рисунок 2.5 – Архітектура MobileNetV2

MobileNetV2, зазвичай, має близько 3,4 мільйона параметрів, що робить її істотно менше і ефективніше в порівнянні з більшими моделями, такими як VGG16 або ResNet-50 [31], які мають десятки або навіть сотні мільйонів параметрів. Що стосується розміру моделі на диску, то він може змінюватись в залежності від способу збереження (наприклад, з використанням стиснення

або без) і від того, включені в збережену модель тільки ваги або структура моделі. Однак, приблизний розмір файлу ваг для MobileNetV2 може бути в межах десятків МБ.

MobileNetV1 була значним досягненням, оскільки запропонувала архітектуру, оптимізовану для мобільних пристроїв, з низькою кількістю параметрів та ефективним використанням обчислювальних ресурсів, зберігаючи при цьому гідну якість розпізнавання. MobileNetV1 використовувала глибокі згортки, що розділяються, для зниження кількості параметрів і обчислювального навантаження.MobileNetV2 – це полегшена архітектура DNN, яка була представлена Google і орієнтована на мобільні платформи та процесори з низькими можливостями обробки. MobileNetV2 є одним із найефективніших DNN з точки зору точності та розміру моделі, і тому вона вважається ідеальною для додатків IoT.

Вона складається з 2-вимірного згорткового шару, вузького місця та об’єднання. Рівень вузького місця, у свою чергу, складається з підрівнів розширення, нормалізації, активації, додавання та згортки 3×3. Метою операції розширення є розширення простору даних за рахунок збільшення кількості каналів вхідних даних на коефіцієнт, який визначається гіперпараметрами моделі. Навпаки, операції проектування зменшують кількість каналів вхідних даних і, таким чином, вони звужують простір даних. Рівні нормалізації, об’єднання, згортки та активації використовують відповідні операції, необхідні для навчання DNN і процесу логічного висновку. Процедура навчання DNN вимагає прямого проходження даних через нейронну мережу, за яким слідує операція зворотного поширення, яка точно налаштовує ваги моделі відповідно до значення функції втрат. Навпаки, висновок DNN не включає жодного процесу налаштування ваги моделі, і, отже, вимагає, щоб вхідні дані передавалися через мережу один раз. Як результат, процес висновку DNN вважається менш витратним з точки зору обчислювальних вимог і споживання енергії порівняно з навчанням DNN. Якщо говорити точніше, процес висновку навченої моделі MobileNetV2

вимагає 13 мільйонів операцій MAC і містить 4,3 мільйона параметрів. Як наслідок, проектування та реалізація операції MAC відіграє вирішальну роль у часі виконання висновку MobileNetV2.

Сьогодні DNN продемонстрували надзвичайний успіх ціною величезної кількості параметрів і обчислень MAC. Однак методи, засновані на скороченні ваги, були створені як привабливий підхід до зменшення вимог до обчислень і пам'яті моделей DNN без шкоди для значних рівнів точності [31]. У цій роботі використовується метод структурованої обрізки на основі [32] у двох режимах.

По-перше, використовується консервативний поріг для підтримки високої точності MobileNetV2.

По-друге, для порівняння з базовими моделями та консервативним підходом використовується більш агресивний підхід скорочення.

Варто зазначити, що на відміну від методів неструктурованого скорочення, методи структурованого скорочення зазвичай є більш дружніми до апаратного забезпечення методологіями, оскільки вони забезпечують більшу регулярність, а також можуть досягти порівнянних показників скорочення.

Надалі оцінюємо всі підходи, використовуючи стислі формати стовпців і рядків, щоб мінімізувати вимоги до пам'яті та ефективно прискорити швидкість використання даних для операцій MobileNetV2. У межах кожного підходу апаратні ядра однакового дизайну кодуються за допомогою подібних стиснутих форматів.

Таким чином, використовується загальна схема стиснення, що зменшує нерегулярність розріджених ваг. MobileNetV2 пішла далі MobileNetV1 та внесла кілька ключових інновацій для покращення продуктивності.

1. Inverted Residual Blocks. У MobileNetV2 був представлений новий тип блоку, який використовує residual connections навколо Lightweight Depthwise Convolutions. Це дозволяє мережі навчатися ефективніше та покращує передачу інформації між шарами мережі.

2. Linear Bottlenecks. У цих блоках внутрішнє подання даних стискається таким чином, щоб активації залишалися лінійними. Це допомагає запобігти втраті інформації під час перетворення.

3. Revisited Depthwise Separable Convolutions. MobileNetV2 також оптимізує шари глибоких згорток, що розділяються, які були введені в першій версії MobileNet.

Ці нововведення дозволили MobileNetV2 досягти кращої точності, ніж її попередниця, при одночасному зниженні кількості операцій та ваг. Це робить її ще більш придатною для пристроїв з обмеженими обчислювальними ресурсами, таких як смартфони, планшети та системи, що вбудовуються.

MobileNetV2 та її попередники мали великий вплив на область комп'ютерного зору та машинного навчання, оскільки вони дозволяють розробляти потужні та ефективні моделі для реальних додатків, де обчислювальні ресурси дуже обмежені.

MobileNetV2 привнесла кілька важливих переваг у світ комп'ютерного зору та машинного навчання, особливо коли справа стосується впровадження потужних, але ефективних моделей у пристрої з обмеженими обчислювальними ресурсами. Ось ключові переваги MobileNetV2.

1. Висока ефективність. MobileNetV2 була спеціально розроблена для роботи на мобільних пристроях та інших платформах. Вона використовує менше обчислень і пам'яті для обробки даних, що робить її ідеальною для систем, що вбудовуються.

2. Баланс між точністю та продуктивністю. MobileNetV2 забезпечує гарний баланс між обчислювальною ефективністю та точністю моделі. Завдяки нововведенням, таким як inverted residuals та linear bottlenecks, вона досягає високої точності навіть при зниженні кількості операцій та параметрів.

3. Гнучкість. Архітектура MobileNetV2 дозволяє легко масштабувати модель для відповідності різним вимогам та умовам, наприклад, шляхом

зміни ширини шарів (кількість каналів) та роздільної здатності вхідного зображення.

4. Перенесення. Моделі, засновані на MobileNetV2, легко адаптуються та переносяться на різні платформи та пристрої, включаючи мобільні телефони, IoT пристрої та інші системи, де потужність та пропускна здатність пам'яті обмежені.

5. Підтримка вимог реального часу. Завдяки своїй легковажності та ефективності, MobileNetV2 підходить для завдань, що потребують обробки в реальному часі, таких як розпізнавання об'єктів, сегментація та класифікації.

Незважаючи на значні переваги MobileNetV2, існують також певні недоліки та обмеження, які важливо враховувати.

1. Компроміс між продуктивністю та точністю. Хоча MobileNetV2 досягає вражаючого балансу між розміром моделі та точністю, це все одно компроміс. У завданнях, де потрібна максимально можлива точність, більш глибокі та ресурсомісткі архітектури, такі як ResNet або Inception, можуть перевершити MobileNetV2.

2. Складність оптимізації. Інвертовані залишкові з'єднання та лінійні вузькі місця можуть ускладнити оптимізацію мережі порівняно з традиційнішими архітектурами.

3. Залежність специфікацій пристрою. Хоча MobileNetV2 оптимізована для мобільних пристроїв, різні пристрої можуть мати різні характеристики обладнання, що може впливати на продуктивність моделі. Певні оптимізації, необхідні для одних пристроїв, можуть бути неефективними або контрпродуктивними на інших.

4. Вимоги до обладнання для навчання. Хоча сама модель та ефективна в інференсі, процес її навчання все ще потребує потужного обладнання, такого як GPU або TPU, що може бути бар'єром для розробників з обмеженими ресурсами.

5. Невизначеність поведінки при перенесенні навчання. Перенесення навчання з використанням MobileNetV2 може бути менш передбачуваним у

порівнянні з більшими мережами, особливо якщо цільова датасет сильно відрізняється від того, на якому модель спочатку навчалася (наприклад, ImageNet).

6. Обробка складних завдань. Для деяких складних завдань комп'ютерного зору, що вимагають вилучення дуже тонких особливостей даних, MobileNetV2 може не забезпечити потрібної продуктивності в порівнянні з більш глибокими і широкими мережами, здатними вловлювати більш складні патерни.

Ці обмеження означають, що інженери та дослідники повинні ретельно розглянути вимоги своїх додатків при виборі MobileNetV2 і бути готовими до можливості, що деякі завдання потребують більш потужної архітектури. Якщо у попередніх версіях до MobileNetV3 оптимальна архітектура відшукалася вручну, то у 3-ій версії головною первагою стало знаходження архітектури методами MnasNet та NetAdapt (рис. 2.6).

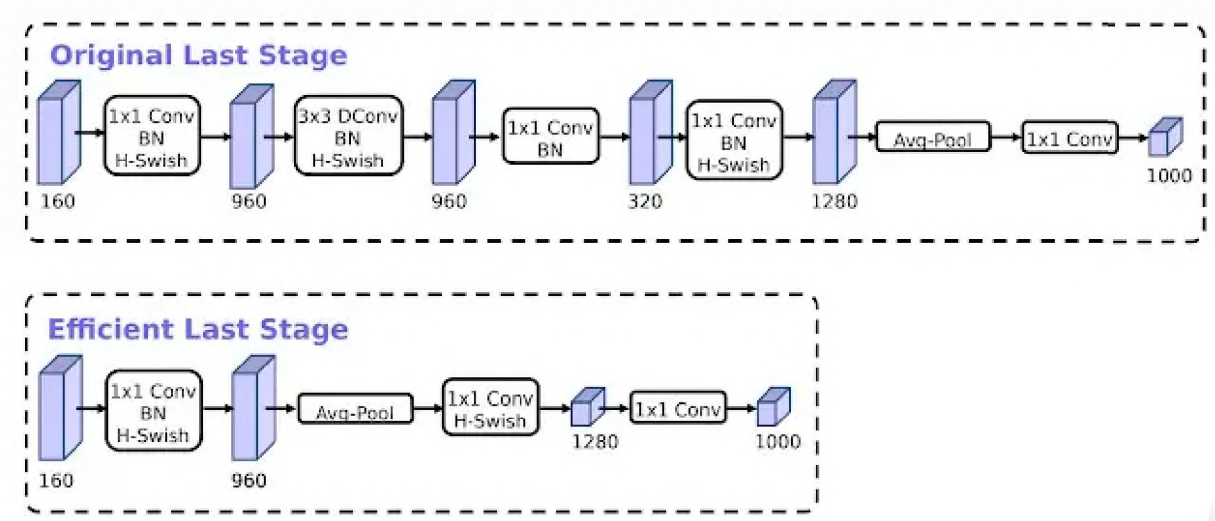

Рисунок 2.6 – Як змінилися фінальні шари після підбору архітектури

Обидва методи розроблені спеціально для пошуку архітектури сетей під мобільні пристрої. Вони враховують час роботи на них для вибору оптимальних за співвідношенням якості/швидкості варіантів. Крім того, в набір додано блоки Squeeze-and-Excitation, що зв'язують канали мережі (рис. 2.7). По-суті, це увага для окремих каналів вхідного тензора:

1. Спочатку стискаємо початковий тензор H×W×C до 1×1×C за допомогою середнього об'єднання, тож є середній кожен канал окремо.

2. Лінійним шаром з сигмоїдною на кінці відображаємо ці C середніх значень у C вагів, по одному на кожному каналі.

3. Останнім кроком розмножуємо кожен канал на відповідне йому число і укладаємо з початковим тензором.

На рис. 2.8 показано, куди цей блок вставили. Зауважимо, що зважуємо саме «товсті» тензори, на яких застосовується нелінійність. Це теж результат пошуку архітектури.

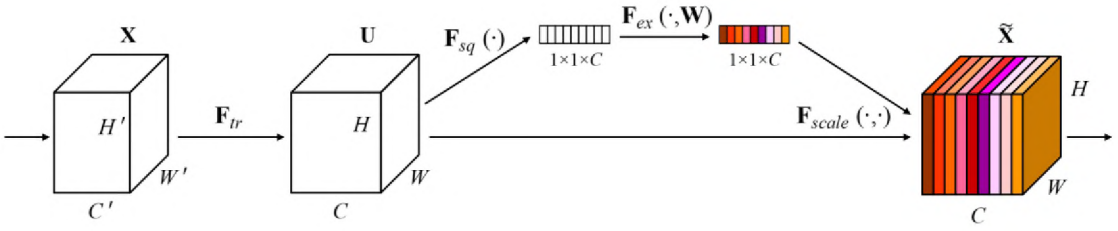

Рисунок 2.7 – Squeeze-and-Excitation

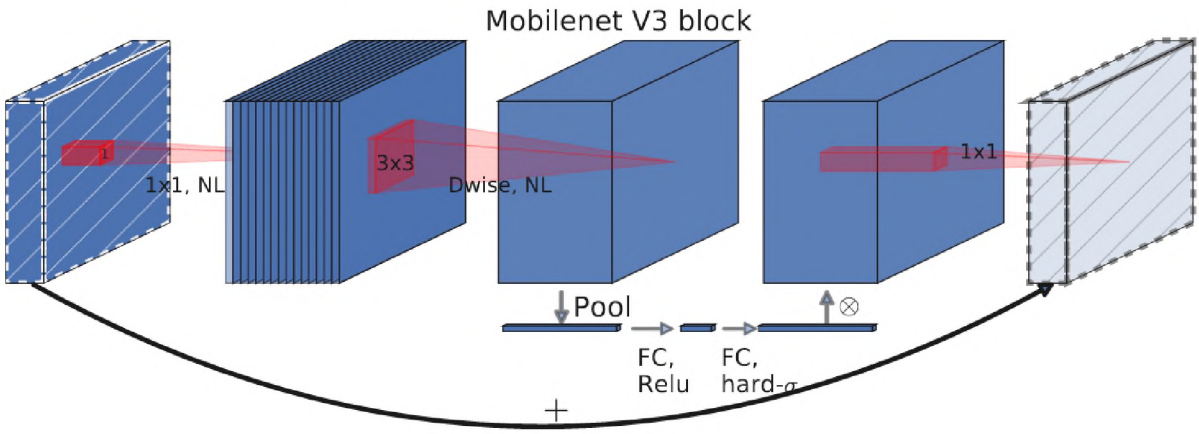

Рисунок 2.8 – Блок MobileNetV3 з використанням Squeeze-and-Excitation

Враховуючі найбільш досліджену версію архітектура MobileNet, надалі в роботі розглядається MobileNetV2 для реалізації сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості.

2.3 Архітектура Xception

Щоб покращити якість розпізнавання своїх мереж, дослідники намагалися додавати в мережі більше шарів, проте згодом прийшло розуміння, що іноді обмеження продуктивності просто не дозволяють навчати та використовувати такі глибокі мережі. Це стало мотивацією для використання depthwise separable convolutions і створення архітектури Xception (скорочення від «Extreme Inception») [34]. Inception (рис. 2.9) базується на гіпотезі, що міжканальні кореляції та просторові кореляції достатньо роз'єднані. При цьому, вперше реалізується міжканальні кореляції через набір згорток 1×1. Потім діє як «багаторівневий екстрактор ознак», обчислюючи згортки 1×1, 3×3 та 5×5.

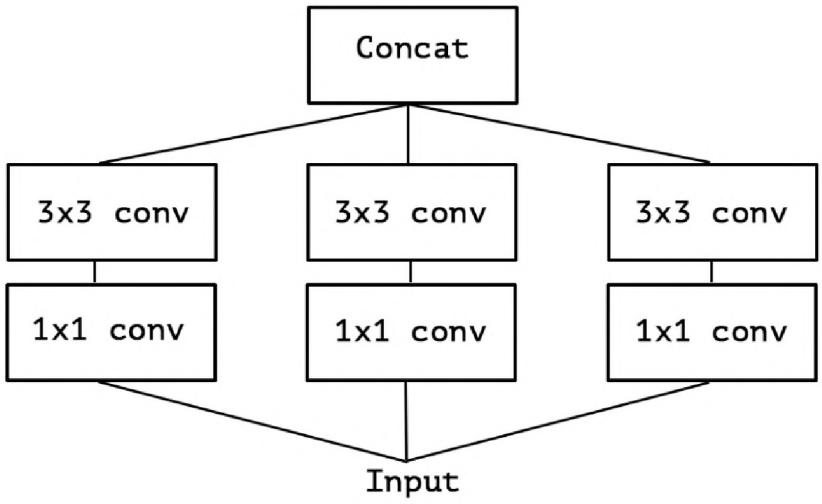

Рисунок 2.9 – Архітектура Inception

Xception є однією з моделей глибокого навчання в галузі CV, яка була введена в 2016 р. Франсуа Шолле, співробітником Google. Xception базується на «екстремальній» інтерпретації моделі Inception від Google (рис. 2.10). Спочатку використовується згортка 1×1 для відображення міжканальних кореляцій. Потім окремо відображається просторова кореляція кожного вихідного каналу (замість лише 3-4 секцій) – подібно до роздільної згортки по глибині. Таким чином, Xception покращує попередню модель Inception V3,

використовуючи підхід, названий «глибоким поділом», який розділяє операції згортки по каналах та просторові операції (рис. 2.11). Основна ідея Xception полягає в тому, що згорткові операції можуть бути ефективно виконані окремо в кожному каналі вхідних даних, а потім послідовно у просторових вимірах, замість того щоб виконувати їх разом. Це дозволяє мережі адаптуватися до специфічних просторових та канальних ознак даних незалежно один від одного.

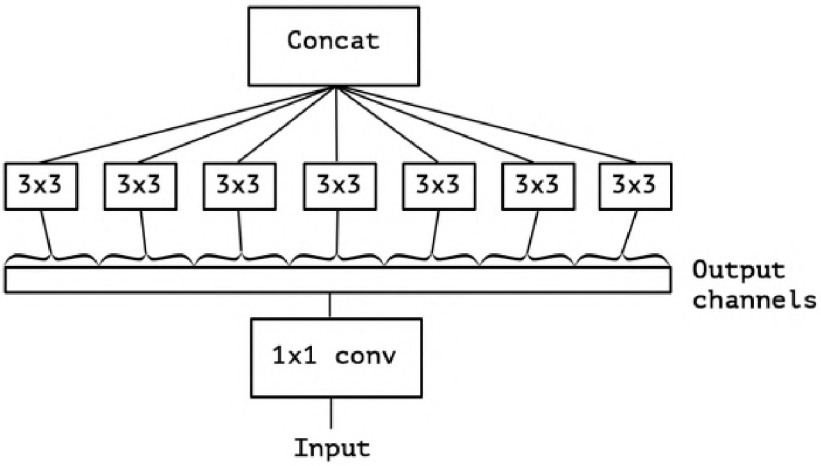

Рисунок 2.10 – «Екстремальна» версія модуля Inception

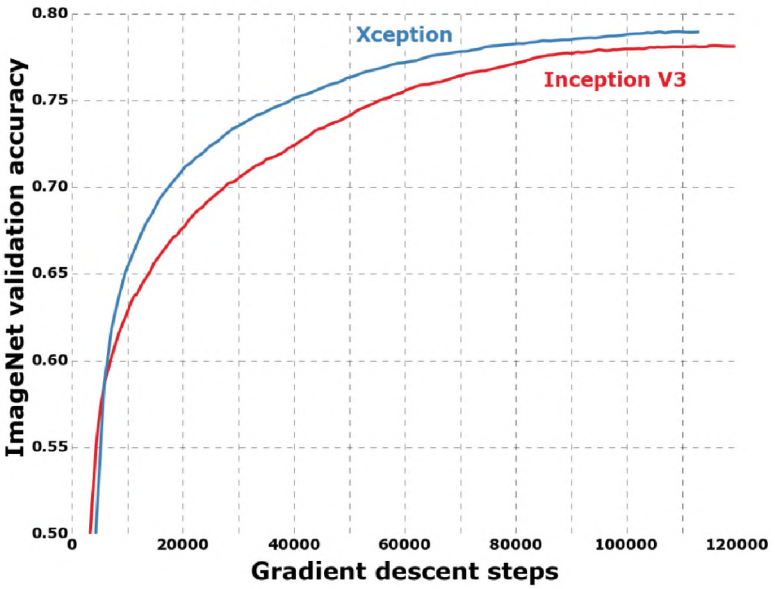

Рисунок 2.11 – Порівняння Xception і Inception v3

Тобто реалізується лінійний стек шарів згортки, які можна розділити по глибині, із залишковими зв'язками. Xception складається з 71 шару і

використовує архітектуру, яка дозволяє масштабуватися для різних розмірів вхідних даних і бути високо ефективною з точки зору обчислень. Тобто реалізується роздільна згортка по глибині, що дозволяє регулярній згортці дивитися на канальні та просторові кореляції одночасно (рис. 2.12):

– просторова згортка – 3×3 згортки для кожного каналу;

– згортка по глибині: згортки 1×1 на зчеплених каналах;

– регулярна згортка: 16×32×3×3 = 4608 параметрів;

– роздільна згортка по глибині: (просторова conv + глибинна conv) = (16×3×3 + 16×32×1×1) = 656 параметрів;

– значно зменшена кількість параметрів;

– більш ефективна комплексність;

– підтримує крос-канальні функції.

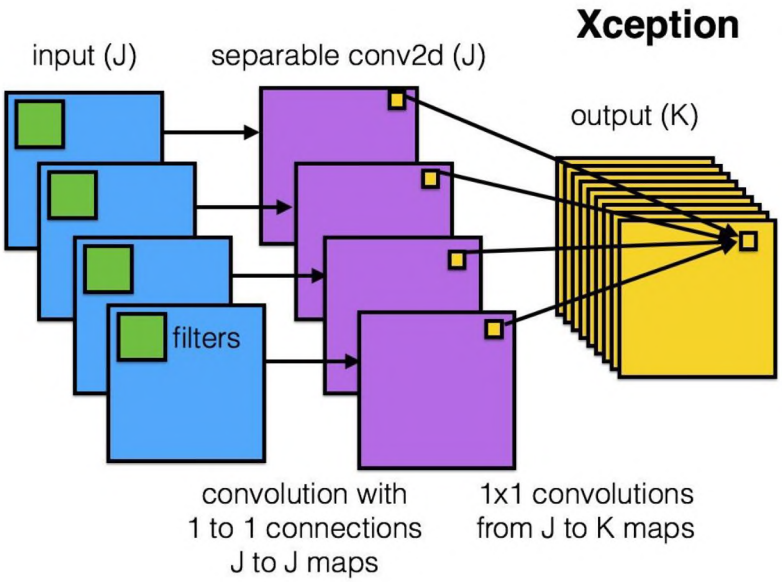

Рисунок 2.12 – Приклад згорткового шару 3×3 на 16 вхідних каналах і 32 вихідних каналах

Це досягається за рахунок використання глибоких згорткових нейронних мереж з залишковими сполуками, що допомагає впоратися з проблемою градієнта, що зникає, при навчанні глибоких мереж. Xception була протестована на наборах даних ImageNet та JFT і показала чудову продуктивність у порівнянні з Inception V3, а також з іншими сучасними

архітектурами того часу, такими як ResNet та VGG. З того часу Xception застосовується у багатьох завданнях CV, включаючи класифікацію зображень, виявлення об'єктів та сегментацію. Архітектура (рис. 2.13) вплинула на розробку наступних моделей згорткових нейронних мереж і на практичне застосування глибокого навчання в індустрії та наукових дослідженнях.

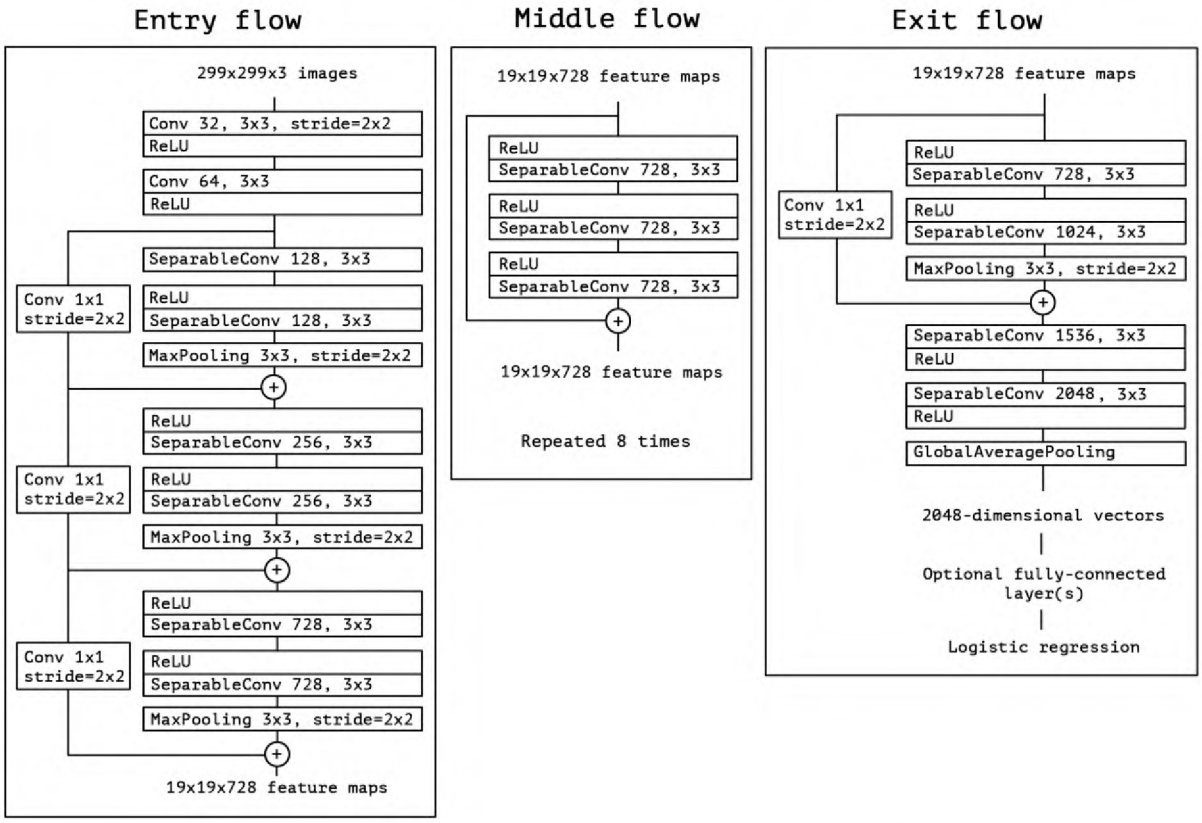

Рисунок 2.13 – Архітектура Xception

Чим відрізняється роздільна згортка по глибині? Дві відмінності між «екстремальною» версією модуля Inception і згорткою, що розділяється по глибині. По-перше, порядок операцій – згортки, що розділяються по глибині, спочатку виконують просторову згортку по каналу, а потім виконують згортку по глибині 1×1 на виході. По-друге, нелінійність після першої операції – на Inception обидві операції супроводжуються нелінійністю ReLU.

Архітектура Xception повністю заснована на шарах згортки, які можна розділити по глибині та базується на фундаментальній гіпотезі: відображення міжканальних кореляцій та просторових кореляцій може бути повністю відокремлено. Xception складається з 36 згорткових шарів, які формують

основу вилучення ознак мережі, структурована на 14 модулів, усі з яких мають лінійні залишкові зв’язки навколо них, за винятком першого та останнього модулів. Архітектура Xception була представлена як значне покращення порівняно з попередніми обертовими нейронними мережами завдяки кільком ключовим перевагам.

1. Покращений розподіл ознак. Xception використовує ідею, що операції зв'язку можуть бути більш ефективними та ефективно розділені на операції через канали, а потім у просторових вимірах. Це означає, що модель може навчитися більш складним і різноманітним ознакам, тому що вона обробляє канальні та просторові властивості по-різному.

2. Остаточне з'єднання (Residual Connections). Введення остаточних з'єднань допомагає вирішити проблему вичерпного градієнта при навчанні дуже глибоких мереж, що полегшують зворотні помилки і покращуючи узагальнюючу здатність мережі.

3. Висока продуктивність. Xception показав чудову продуктивність у багатьох задачах класифікації зображень, оперуючи своїми попередниками, такими як Inception V3, і багато інших сучасних архітектур на момент її представлення.

4. Ефективне використання параметрів. при більш ефективному використанні параметрів у порівнянні з Inception V3, Xception демонструє кращі результати без значного збільшення кількості параметрів, що робить її переважно з точки зору вимірної ефективності.

5. Гнучкість і масштабованість. Архітектура Xception демонструє гнучкість і масштабованість, що дозволяє її використовувати для різних розмірів вхідних даних і в різних задачах комп’ютерного зору, від класифікації до виявлення об’єктів і сегментації.

6. Трансферне навчання. Xception добре підходить для трансферного навчання, де попередньо навчальну модель можна адаптувати для нової, але пов'язаної задачі. Це широко використовується в додатках, де дані обмежені або специфічні для домену.

Ці переваги зробили Xception популярною архітектурою для дослідників і практиків у сфері комп'ютерного зору, де вона часто служить базою для більш складних систем.

Архітектура Xception, хоча і потужна, не позбавлена недоліків і обмежена, подібна будь-якій іншій архітектурі глибокого навчання.

1. Вимоги до вимірних ресурсів. Незважаючи на ефективне використання параметрів у порівнянні з деякими іншими моделями, Xception все ще вимагає значної кількості вимірних ресурсів для навчання та вказівок, особливо у випадку роботи з високорозмірними зображеннями.

2. Складність і час навчання. Через глибину та складність мережі час навчання може бути досить тривалим, особливо без доступу до оптимізованої вичислювальної інфраструктури, такої як GPU або TPU.

3. Перенавчання (Overfitting). Як і в інших глибоких сетях, існує ризик перенавачання, особливо коли доступно мало даних для навчання. У таких випадках модель може вивчити шум із тренувального набору даних, що знижує її здатність до узагальнення нових даних.

4. Оптимізація. Налаштування гіперпараметрів для Xception може бути досить складною задачею. Вона вимагає ретельного підбору швидкості навчання, стратегії регулярізації та інших факторів для досягнення оптимальної продуктивності.

5. Універсальність. Хоча Xception добре працює для завдань обробки зображень, її архітектура може бути не настільки ефективною для інших типів даних і завдань, таких як послідовності тимчасових рядів або природний язик, для яких існують більш спеціалізовані моделі.

6. Оновлення та інновації в області. Як і в будь-якій іншій області досліджень, в області глибокого навчання постійно з'являються нові і більш досконалі архітектури. Таким чином, Xception може заблокувати місце більш нових архітектур, які можуть запропонувати кращу продуктивність або більш ефективне використання ресурсів.

7. Специфіка домену. Попередньо навчальні моделі Xception зазвичай навчаються на великих і загальних наборах даних, таких як ImageNet. Коли вони використовуються в більш специфічних доменних додатках, може знадобитися значна доробка моделей і її перенавчання на нові дані для досягнення прийнятної продуктивності.

Як і будь-яка інша модель, Xception найкраще підходить для використання конкретних сценаріїв і має застосовуватися з урахуванням ваших обмежень і контексту завдань. Архітектура Xception, як і багато інших сучасних світових нейронних мереж, була розроблена з можливістю обробляти зображення різних розмірів. Однак, є рекомендовані розміри зображень, на яких архітектура буде працювати найбільш ефективно, особливо якщо використовується попередньо навчальна на набір даних модель ImageNet. Стандартний розмір вхідного зображення для Xception становить 299×299 пікселів. Цей розмір був обраний у якості вхідного розміру зображення для оригінальної моделі Inception, на основі якої розроблено Xception. Коли ви використовуєте попередньо навчені ваги, вам слід зберегти цей розмір зображення, щоб забезпечити оптимальну продуктивність і уникнути необхідності змінювати вагові моделі.

Однак, якщо ви навчаєте Xception з нуля (тобто без попередньо навчених вагів), ви можете використовувати зображення інших розмірів, за умови, що вони досить великі, щоб пройти через усі шари згорткової мережі, і при цьому залишилися досить маленькими, щоб не перевищити пам'ять вашого обладнання. У цьому випадку може знадобитися налаштування архітектури, наприклад, зміна кількості шарів або фільтрів, щоб адаптувати її під новий розмір вхідних даних. Також важливо відзначити, що багато глибокі навчальні фреймворки автоматично адаптують масу в повнозв'язних шарах залежно від розміру вхідного зображення, що робить використання відмінних розмірів вхідних даних більш гнучким. Тем не менш, завжди рекомендується перевіряти специфікації конкретної реалізації Xception, з

якою працюєте. Щоб реалізувати архітектуру Xception, можна скористатись кодом, що наведений у Додатку А.

Надалі доцільно використовувати Transfer Learning для реалізації проєкту Edge Computing + AI CV. Це метод у машинному навчанні, де модель, навчена на одному завданні, використовується як відправна точка для іншої, пов'язаної задачі. Це використовувати знання, отримані в одному предметі, наприклад в математиці, щоб легше вчити фізику. Це заощаджує час та ресурси, тому що не потрібно навчати модель з нуля.

Висновки до розділу 2

Сезонно-специфічний сегмент багаторівневої архітектури нейронної мережі класифікації зображень місцевості містить два блоки. Перший блок проводить класифікацію сезону, що відповідає зображенню.

Для синтезу першого блоку в роботі запропоновано дві альтернативні архітектури MobileNetV2 і Xception. Такий підхід орієнтований на реалізацію проєктів Edge Computing + AI + CV.

Виконано дослідження запропонованих CNN, та розглянуті особливості реалізації моделі глибокого навчання нейронної мережі класифікації сезону з використанням Transfer Learning.

У другому блоці відповідно до сезону вирішується основне задання. Для виконання основне задання CV другий блок базується на архітектурі одні з версій YOLOv8.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО РЕАЛІЗАЦІЇ СЕЗОННО-СПЕЦИФІЧНОГО СЕГМЕНТУ БАГАТОРІВНЕВОЇ АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ МІСЦЕВОСТІ

3.1 Програмно-апаратні компоненти для оцінки моделі глибокого навчання синтезованої нейронної мережі

Одним з компонентів сезонно-специфічний сегмент багаторівневої архітектури нейронної мережі класифікації зображень місцевості є модель глибокого навчання класифікацію сезону [34]. Згідно п. 2.1, для її реалізації доцільно використовувати архітектуру MobileNet або Xception. З метою визначення пріоритетності вибору одного з вказаних варіантів моделі потрібно провести оцінку продуктивності.

Набір зображень поверхні землі зі снігом та без нього є специфічним для мереж MobileNet та Xception. Тому необхідно виконувати донавчання даних мереж. Для цього можна скористатись хмарним сервісом Google Colaboratory [35] на мові Python, який досить вдало підходить для проєктів на основі бібліотеки Keras [36]. Використовуючи певний план оплати даного сервісу в процесі навчання нейронної мережі можна використовувати серверну відеокарту A100-SXM4-40GB (рис. 3.1) [37].

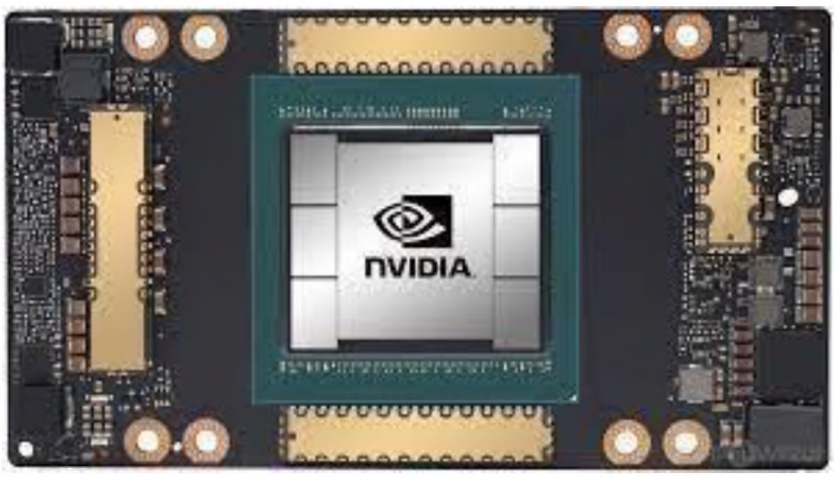

Рисунок 3.1 – A100-SXM4-40GB

Вона була випущена к. NVIDIA у травні 2020 р. Відеокарта побудована на архітектурі Ampere з кодовою назвою GA100 за 7 нм технологічним процесом і містить 54200 мільйонів транзисторів. Відеокарта споживає 400 Вт. Частота ядра складає 1095 МГц, а в режимі Boost – 1410 МГц. Кількість шейдерних процесорів дорівнює 6912. Швидкість текстурування – 609,1 GTexel/c. Максимальний розмір пам'яті – 40 ГБ типу HBM2e. Ширина шини пам'яті – 5120 бітів. Частота пам'яті – 1215 МГц (2,4 Гбіт/с). Дана відеокарта містить CUDA-ядра з одинарною точністю та 432 тензорних ядра/GPU і реалізує швидкість 19,5 TFlops. На даний час, в ЦОД використовують збірки відеокарт. Наприклад, на рис. 3.2 наведено Redstone, що містить 4x A100 40 ГБ SXM4 [38]. Все це дозволяє дослідникам прискорювати програми та працювати з ще більшими моделями та наборами даних. На рис. 3.3 наведено середнє навантаження на апаратний сегмент під час проведення досліджень запропонованих моделей в Google Colaboratory.

Рисунок 3.2 – NVIDIA Redstone GPU Baseboard

Рисунок 3.3 – Навантаження під час навчання нейронної мережі

При цьому, змінювались гіперпараметри, що стосувались процесу навчання нейронної мережі. В першу чергу це стосується розміру батча і кількості епох. В загальному випадку, розглядались архітектури, що мають продуктивність не менше 85 %. В якості метрики для оцінки моделей вибрано *Balanced Recall* [39]. Дана метрика дорівнює середньому арифметичному *Recall* (або точності) для кожного класу:

$$Balanced\ Recall = \frac{1}{2}(Recall_{позитивний\ клас} + Recall_{негативний\ клас}). \quad (3.1)$$

Враховуючі різні вимоги Xception і MobileNet до розміру зображень на вході нейронної мережі, розроблені моделі глибокого навчання виконаю трансформацію зображень.

База містить зображення двох класів: поверхня землі зі снігом (клас SNOW) – 154 шт. та без нього (клас NOT_SNOW) – 196 шт. Зображення робились з борту пасажирського літака. Сам датасет формується у пропорції 70 % – на навчальну вибірку (рис. 3.4) та 30 % – на перевірочну (SNOW – 47 шт. і NOT_SNOW – 59 шт.

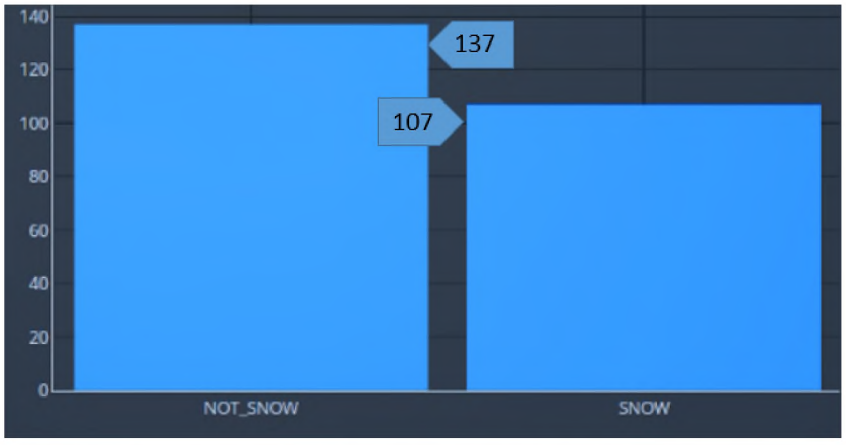

Рисунок 3.4 – Навчальна вибірка датасету

Тестова вибірка містила 27 зображень, що не перетиналися з навчальною та перевірочною вибірками.

3.2 Модель на основі MobileNetV2

Для класифікації зображень на основі сформованого датасету необхідно підключити вибрану архітектуру. Згідно [40], для нейронної мережі MobileNetV2 фрагмент шаблону коду має вигляд:

```
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model

input_1 = Input(shape=(180, 240, 3), name='1')
x_3 = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True,
scale=True, beta_initializer='zeros', gamma_initializer='ones',
moving_mean_initializer='zeros', moving_variance_initializer='ones',
beta_regularizer='l1_l2', gamma_regularizer='l1_l2', beta_constraint='min_max_norm',
gamma_constraint='min_max_norm', name='BatchNormalization_3')(input_1)

x_16 = MobileNetV2(include_top=False, weights='imagenet', pooling='avg', classes=1000,
classifier_activation='softmax', alpha=1.0)
for layer in x_16.layers:
    layer.trainable = False
x_16 = x_16(x_3)
```

В роботі розглядається архітектура, що наведена на рис. 3.5. Вона містить $\approx 2{,}4 \times 10^6$ параметрів (рис. 3.6).

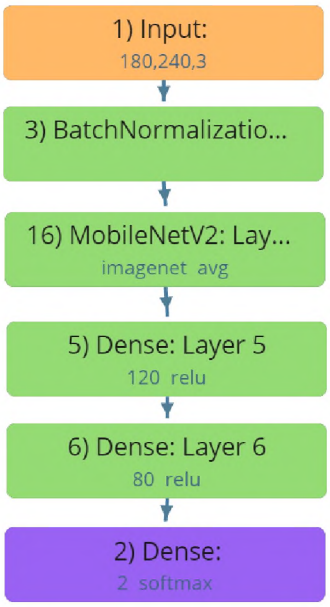

Рисунок 3.5 – Архітектура на мережі на основі MobileNetV2

Для даної моделі виконувалась оптимізація гіперпаратмерів. Спочатку був вибраний розмір батчу – 32, крок навчання дорівнює 0,001. При вибраних

значеннях спостерігалось перенавчання після 40 епох. Тому для даної комбінації обмежились цією кількістю епох.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 1 (InputLayer)              [(None, 180, 240, 3)]     0

 BatchNormalization_3 (Batch  (None, 180, 240, 3)      12
 Normalization)

 mobilenetv2_1.00_224 (Funct  (None, 1280)             2257984
 ional)

 MobileNetV2_16 (Activation)  (None, 1280)             0

 Dense_5 (Dense)             (None, 120)               153720

 Dense_6 (Dense)             (None, 80)                9680

 2 (Dense)                   (None, 2)                 162

=================================================================
Total params: 2,421,558
Trainable params: 2,387,440
Non-trainable params: 34,118
```

Рисунок 3.6 – Характеристика мережі на основі MobileNetV2

При цьому продуктивність моделі дорівнює на навчальній вибірці – 99,3 %, на перевірочній – 89,2 % (рис. 3.7). Графік помилки навчання наведений на рис. 3.8 (кращий результат на навчальній вибірці – 0,0115, перевірочній – 0,55. Фрагмент матриці помилок при роботі по тестовій вибірці наведений на рис. 3.9.

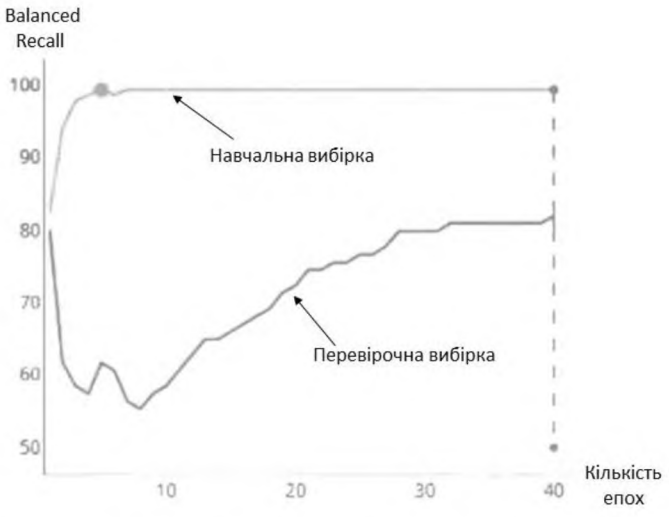

Рисунок 3.7 – Продуктивність мережі на основі MobileNetV2

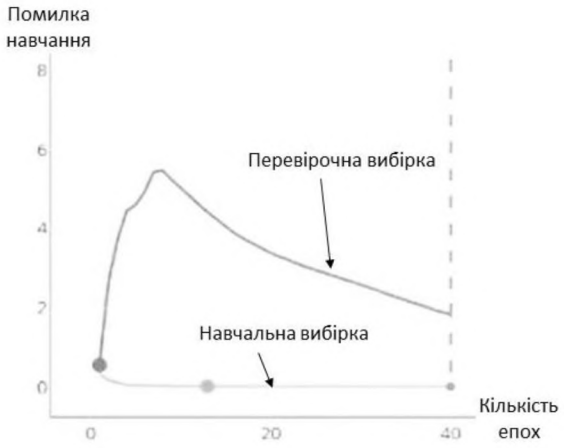

Рисунок 3.8 – Помилка навчання

Наступним етапом була зміна розміру батча (складає 8) та кроку навчання (дорівнює 0,0001). Зменшення батча дозволило знизити навантаження на апаратний сегмент (рис. 3.10).

Шар	Вхідні дані	Вихідні дані	Справжнє значення	Статистика прикладів, %	
	Вхідний шар	Вихідний шар			
	Зображення	Клас	Клас	NOT_SNOW	SNOW
1		SNOW	SNOW	0	100
2		SNOW	SNOW	0	100
3		SNOW	SNOW	0	100
4		SNOW	SNOW	0	100
5		**SNOW**	**NOT_SNOW**	**0**	**100**
6		**NOT_SNOW**	**SNOW**	**1,6**	**98,4**

Рисунок 3.9 – Фрагмент матриці помилок

Рисунок 3.10 – Навантаження при батчі, розмір якого дорівнює 8

Це дозволило отримати продуктивність 100 % навчальній та 93,2 % на перевірочній вибірках (рис. 3.11). Помилка навчання на перевірочній виборці складає 0,0227 (рис. 3.12).

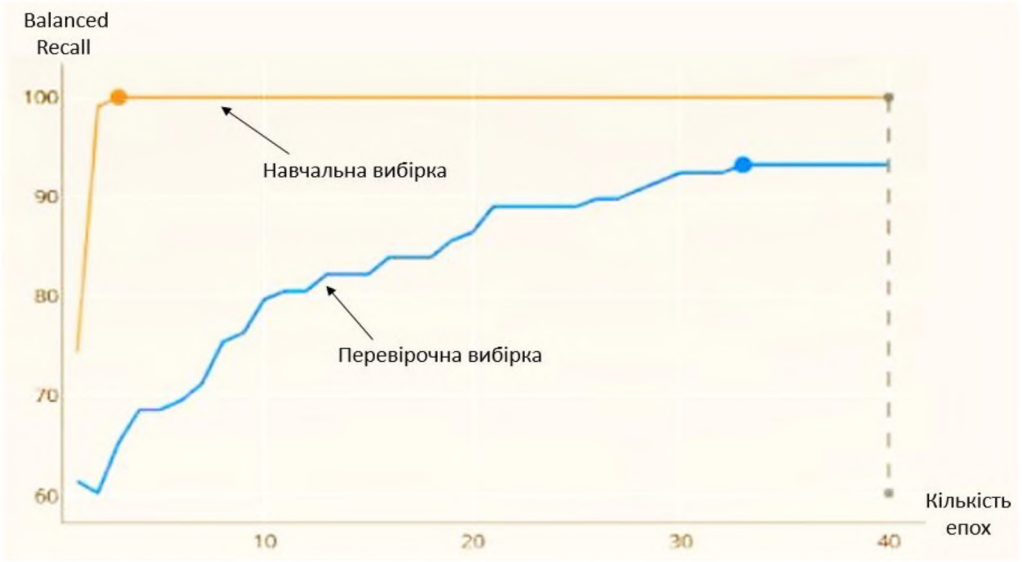

Рисунок 3.11 – Продуктивність мережі на основі MobileNetV2

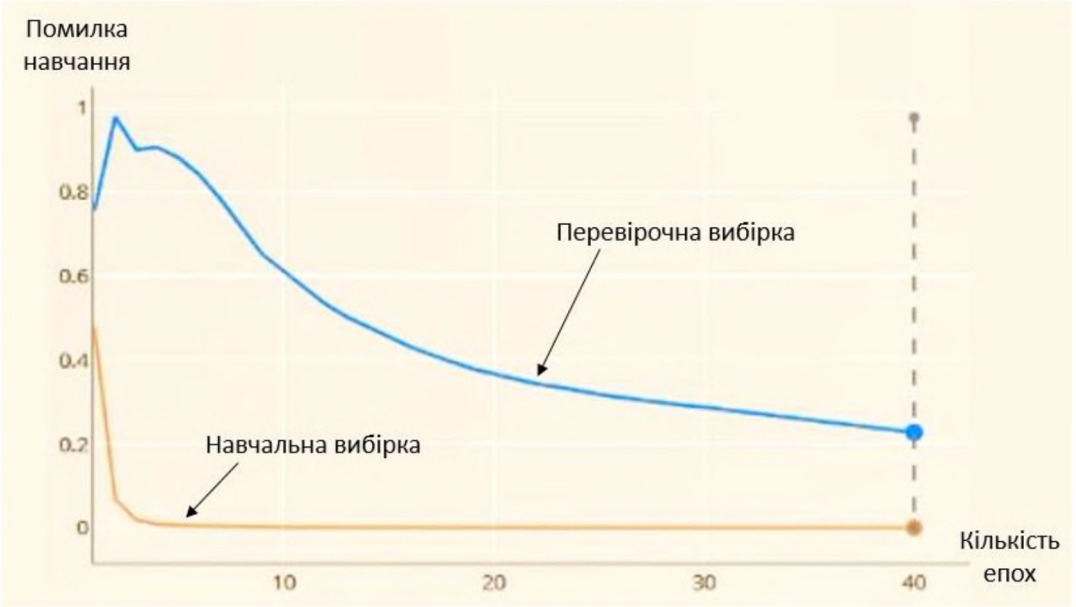

Рисунок 3.12 – Помилка навчання

Матриця помилок наведена на рис. 3.13. Приклад роботи мережі наведено на рис. 3.14. Враховуючі поведінку *Balanced Recall*, можливо збільшити кількість епох навчання з метою підвищення продуктивності. Подальші дослідження дозволили отримати *Balanced Recall* = 96,2 % при

кількості – 75 (рис. 3.15), помилка навчання на перевірочній виборці дорівнює 0,1 (рис. 3.16).

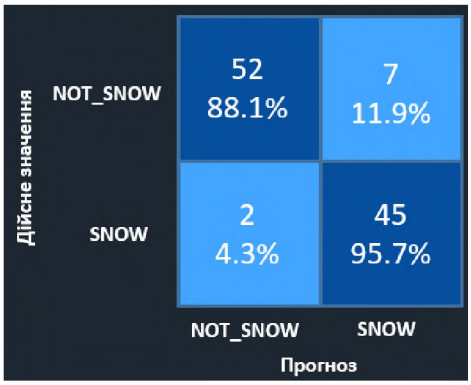

Рисунок 3.13 – Матриця помилок

Рисунок 3.14 – Приклад роботи мережі

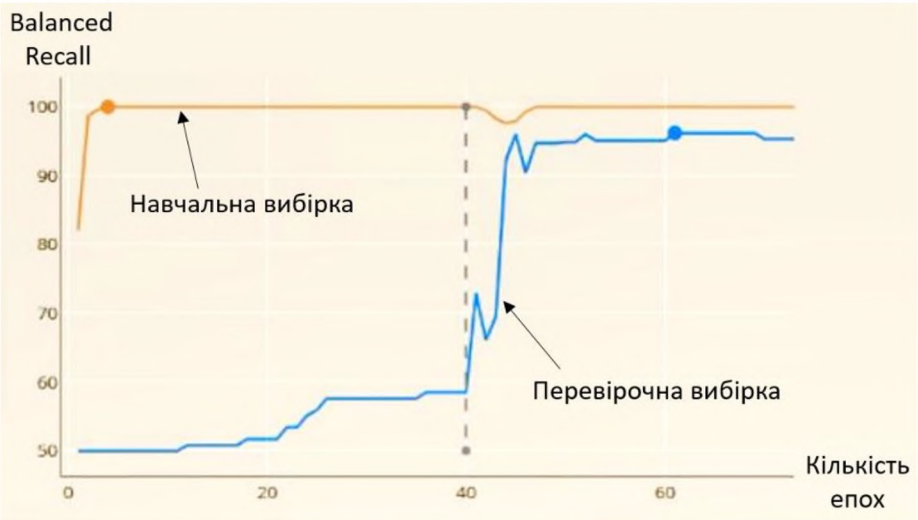

Рисунок 3.15 – Продуктивність мережі

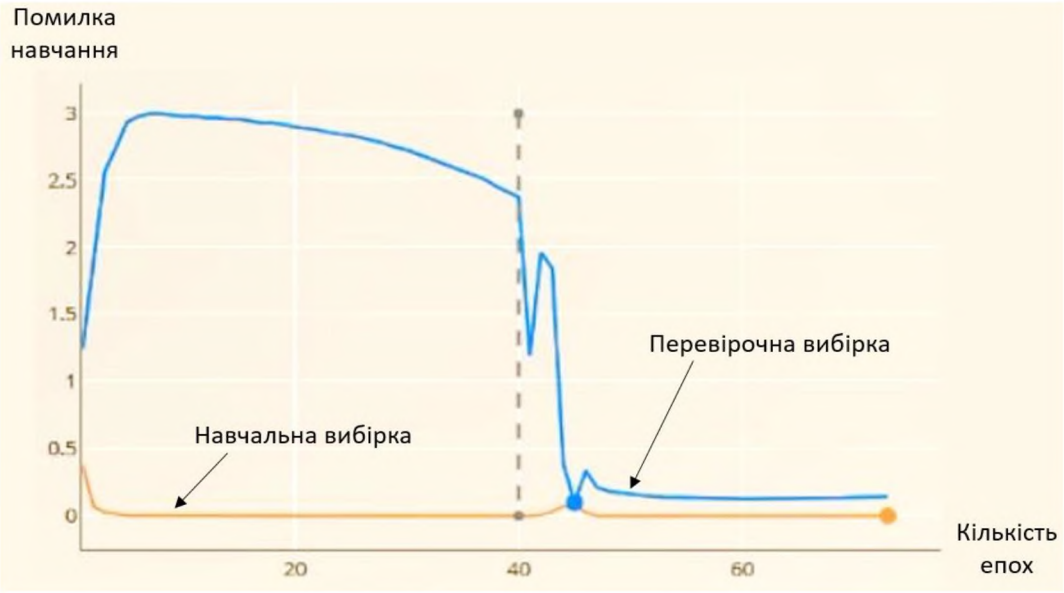

Рисунок 3.16 – Помилка навчання

Проведені дослідження свідчать про можливість використання MobileNetV2 для використання при створенні сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості.

3.3 Модель на основі Xception

Надалі розглянемо використання архітектури Xception [41]. Щоб створити екземпляр архітектури Xception можливо застосувати в Google Colaboratory наступний код:

```
[ ]  tf.keras.applications.Xception(
        include_top=True,
        weights="imagenet",
        input_tensor=None,
        input_shape=None,
        pooling=None,
        classes=1000,
        classifier_activation="softmax",
    )
```

Стандартний розмір вхідного зображення для цієї моделі глибокого навчання становить 299 × 299 пікселей.

При цьому використовуються наступні аргументи:

– include_top – включати повністю підключений рівень у верхню частину мережі.

– Weights – один з None (випадкова ініціалізація), imagenet (попереднє навчання в ImageNet) або шлях до файлу ваги для завантаження.

– input_tensor – необов'язковий тензор Keras (тобто виведення layers.Input()), який використовуватиметься як вхідні дані зображення для моделі.

– input_shape – необов'язковий кортеж форми, вказується тільки в тому випадку, якщо include_top має значення False (інакше вхідна форма повинна бути (299, 299, 3). Вона повинна мати рівно 3 вхідних канали, а ширина і висота повинні бути не менше 71. Наприклад, це (150, 150, 3) буде одне допустиме значення (об'єднання – додатковий режим об'єднання для вилучення функцій, коли include_top є False; None – означає, що вихідні дані моделі будуть вихідними даними 4D-тензора останнього згорткового блоку; avg – означає, що до вихідних даних останнього згорткового блоку буде застосовано глобальне середнє об'єднання, і, таким чином, вихідні дані моделі будуть двомірним тензором; max означає, що буде застосований глобальний максимальний пул).

– класи – необов'язкова кількість класів для класифікації зображень, вказується тільки в тому випадку, якщо встановлено include_top значення True і аргумент не вказано.

– classifier_activation – A str або викликаний. Функція активації для використання на верхньому шарі. Ігнорується, якщо include_top=True. Встановіть classifier_activation=None для повернення логів верхнього шару. При завантаженні попередньо підготовлених ваг classifier_activation може бути тільки Noneабо «softmax».

Надалі було виконано моделювання нейронної мережі. В роботі розглядається архітектура, що наведена на рис. 3.17. Вона містить 21117214 параметрів (рис. 3.18).

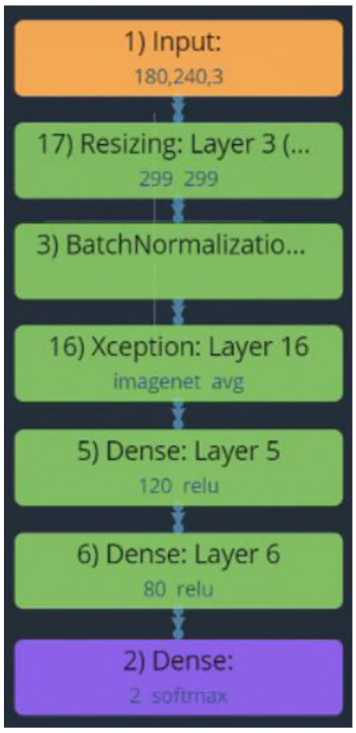

Рисунок 3.17 – Архітектура на мережі на основі Xception

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 1 (InputLayer)              [(None, 180, 240, 3)]     0

 Resizing_17 (Resizing)      (None, 299, 299, 3)       0

 BatchNormalization_3 (Batch  (None, 299, 299, 3)      12
 Normalization)

 xception (Functional)       (None, 2048)              20861480

 Xception_16 (Activation)    (None, 2048)              0

 Dense_5 (Dense)             (None, 120)               245880

 Dense_6 (Dense)             (None, 80)                9680

 2 (Dense)                   (None, 2)                 162

=================================================================
Total params: 21,117,214
Trainable params: 21,062,680
Non-trainable params: 54,534
```

Рисунок 3.18 – Характеристика мережі на основі Xception

При цьому вимого щодо апаратного компоненту реалізації коду Google Colaboratory в процесі навчання нейронної мережі дещо знижуються (рис. 3.19). В моделі було прийняті гіперпараметри, що використовувались в попередній моделі для MobileNetV2.

Рисунок 3.19 – Навантаження в процесі навчання нейронної мережі на основі Xception

При таких значеннях гіперпараметрів отримано продуктивність на рівні 100 % для навчальної і 97,2 % для перевірочної вибірок (рис. 3.20). Помилка навчання на перевірочній виборці склала 0,136 9 (рис. 3.21).

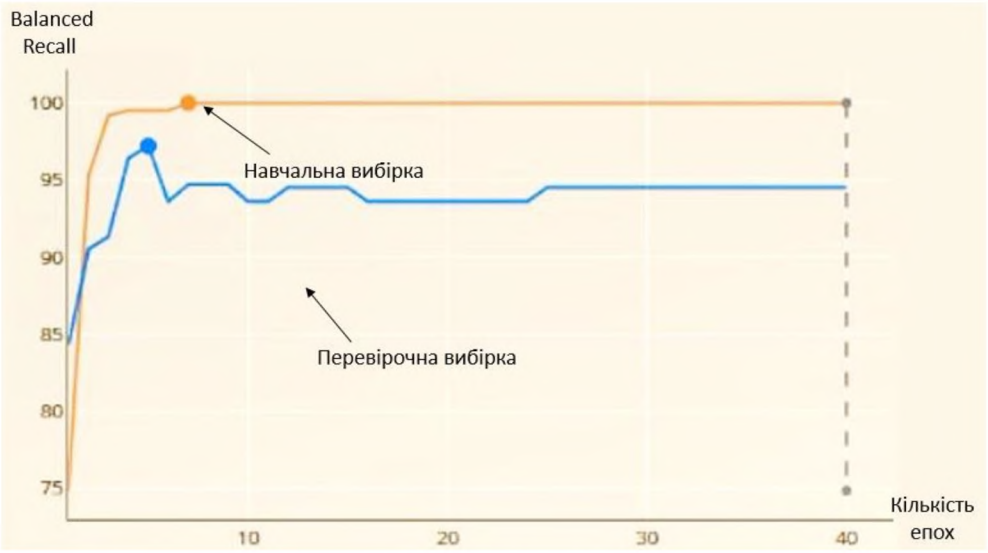

Рисунок 3.20 – Продуктивність мережі на основі Xception

Покращення точності моделі може вимагати множини експериментів та ретельного налаштування. Важливо також стежити за процесом навчання та моніторити метрики, щоб розуміти, які зміни дійсно допомагають досягти кращих результатів для вашого конкретного завдання. Наприклад, можна спробувати методи постобробки результатів моделі, такі як об'єднання декількох моделей або використання порогів для класифікації, додавання додаткових шарів або зміна параметрів мережі.

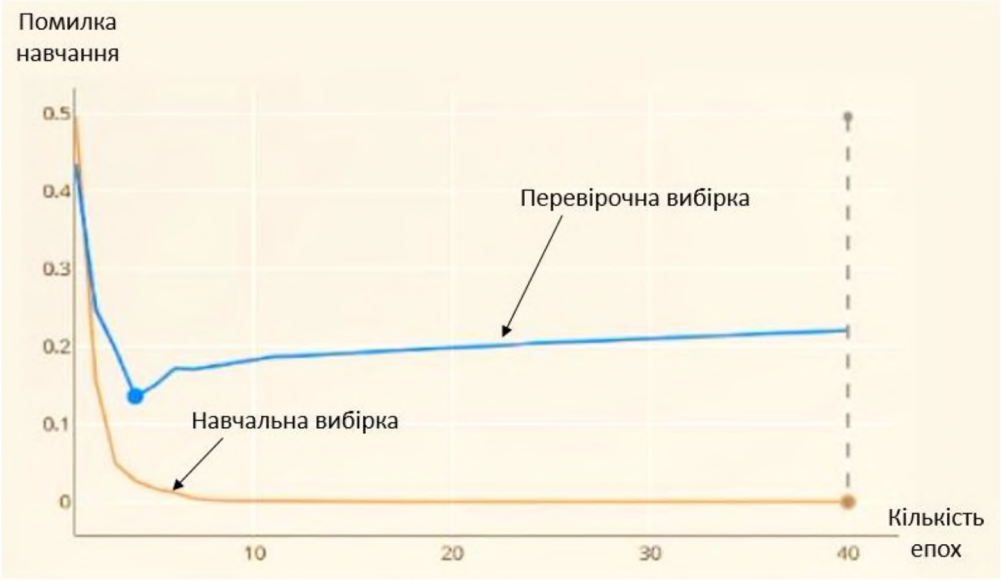

Рисунок 3.21 – Помилка навчання

Також можна застосувати методи регуляризації, такі як dropout та L1/L2 регуляризація [42], щоб зменшити перенавчання моделі. Спробувати зібрати більше навчальних даних та/або застосувати аугментацію даних, щоб збільшити різноманітність тренувальних даних. Це може включати випадкові зміни розміру, повороти, відображення та інші перетворення зображень.

3.4 Обґрунтування вибору нейронної мережі

Xception і MobileNet є архітектурами згорткових нейронних мереж (CNN), але оптимізовані для різних сценаріїв використання. Вибір моделі глибокого навчання між нейронними мережами Xception і MobileNet залежить від вимог до завдання, доступних обчислювальних ресурсів та специфіки застосування, а також кількох чинників, наприклад, необхідна точність, обмеження швидкості обробки та ін. На основі проведених досліджень виконано порівняння, яке може допомогти у виборі між цими двома архітектурами. MobileNetV2 призначена для використання в умовах обмежених обчислювальних ресурсів, таких як мобільні пристрої або

системи, що вбудовуються. Ця архітектура ідеально підходить для додатків, які потребують легковажних моделей, швидкого виведення та ефективності щодо енергоспоживання. MobileNetV2 також може бути корисним при роботі з програмами в реальному часі, де швидкість важливіша, ніж крайня точність, наприклад, у додатках для смартфонів, дронів або IoT-пристроїв.

Xception, зазвичай, краще, коли висока точність класифікації має першорядне значення і доступні достатні обчислювальні ресурси для навчання та інференції. Під терміном «Інференція» розуміють процес використання навченої моделі нейронної мережі для отримання передбачень на нових даних. Тобто, нейронна мережа застосовує вивчені при навчанні ваги та функції до нових даних для вирішення практичних завдань, таких як класифікація зображень, розпізнавання мови, переклад тексту та ін. Процес інференції проходить кілька етапів:

– подача даних – вхідні дані (наприклад, зображення, звук, текст) подаються на вхід моделі;

– пряме поширення (Forward Propagation) – дані проходять через шари нейронної мережі, де кожен нейрон підсумовує зважені вхідні сигнали, застосовує до них функцію активації та передає результати на наступний шар;

– обробка даних – у процесі прямого розповсюдження мережа може застосовувати різні операції (згортка, пулінг, нормалізація та ін. для отримання ознак і зменшення розмірності);

– отримання виводу – на вихідному шарі нейронної мережі формується кінцева відповідь, яка, зазвичай, є ймовірністю приналежності даних до певного класу або конкретний вектор значень (наприклад, координати об'єкта на зображенні для завдань Object Detection);

– постобробка – у деяких випадках вихідні дані моделі вимагають додаткової обробки, наприклад, вибору максимальної ймовірності для класифікації або перетворення ймовірностей на конкретні мітки класів.

Інференція відрізняється від навчання (тренування) тим, що під час навчання модель активно коригує свої ваги на основі помилки передбачення,

використовуючи алгоритми зворотного розповсюдження помилки та оптимізації, тоді як під час інференції ваги залишаються незмінними, і модель просто застосовується для передбачень. Ефективність інференції є критично важливою для розгортання моделей у реальних додатках, де потрібна висока швидкість та ефективність роботи, особливо в завданнях, що потребують роботи в режимі реального часу.

Таким чином, Xception добре підходить для завдань, де потрібна обробка високоякісних зображень і коли розмір та швидкість моделі не є критичними обмеженнями. Вона також підходить для трансферного навчання, коли базова модель навчається на великому та різноманітному наборі даних, а потім адаптується до більш специфічних завдань.

Застосування Xception в рамках Edge Computing може бути не так поширене, як використання більш легких архітектур типу MobileNetV2, зважаючи на вищі вимоги Xception до обчислювальних ресурсів. Тим не менш, у певних сценаріях Xception може бути дуже актуальна, особливо коли необхідно досягти високої точності в складних завданнях комп'ютерного зору і доступні відносно потужні пристрої на краю мережі. Є областей, де застосування Xception у рамках Edge Computing може бути актуальним.

1. Удосконалені пристрої Edge Computing. З розвитком мікропроцесорних технологій деякі пристрої на краю мережі тепер здатні виконувати більш складні обчислення. Наприклад, промислові ПК або спеціалізовані сервери Edge можуть використовувати Xception для більш точного аналізу зображень у реальному часі.

2. Промислове застосування. В якості частини системи машинного зору на виробництві, Xception може допомогти у складній задачі дефектоскопії, де потрібна висока точність для ідентифікації дрібних дефектів на деталях або складальних одиницях.

3. Медичні пристрої та діагностика. У медичних додатках, де надзвичайно важлива точність аналізу зображень (наприклад, при скануванні МРТ або КТ), Xception може використовуватися для попередньої обробки та

аналізу даних на пристроях, що знаходяться безпосередньо у лікарнях або діагностичних центрах.

4. Автономні транспортні засоби. В області самоврядних автомобілів Xception може застосовуватися для розширеного розпізнавання зображень та обробки відеопотоку, щоб точно ідентифікувати об'єкти на дорозі та покращити прийняття рішень у складних умовах довкілля.

5. Безпека та моніторинг. У системах безпеки, де потрібна висока точність у розпізнаванні осіб або інших біометричних даних, Xception може використовуватися для підвищення точності та надійності систем на краю мережі.

Для ефективного застосування Xception в Edge Computing необхідно враховувати не тільки потреби в обчислювальній потужності, але й обмеження, пов'язані зі споживанням енергії та тепловиділенням пристроїв на краю мережі. У деяких випадках може знадобитися спеціалізоване апаратне забезпечення, наприклад GPU або TPU, для обробки моделі в реальному часі. Також варто враховувати можливості кешування та передобробки даних на пристроях краю мережі, щоб мінімізувати кількість звернень до потужніших хмарних серверів. Це може дозволити використовувати Xception в Edge Computing більш ефективно, поєднуючи переваги глибокого навчання.

Надалі розглянемо деякі ситуації використання моделі глибокого навчання. Якщо проект має справу з веб-сервісами, хмарними обчисленнями або потужними серверами без строгих обмежень швидкості обробки, Xception може бути хорошим вибором. Якщо додаток вимагає впровадження Edge Computing, де ефективність та здатність до швидкої інференції більш важливі, MobileNetV2, ймовірно, буде більш підходящим вибором. Обидві моделі можуть бути попередньо навчені на великих наборах даних, таких як ImageNet, а потім налаштовані з використанням тонкого налаштування (fine-tuning) для більш специфічних завдань. Таким чином, вибір між Xception і

MobileNetV2 багато в чому залежить від компромісу між точністю та ефективністю, який найкраще відповідає вашим потребам та обмеженням.

В свою чергу, виникає питання вибору між Xception і MobileNetV3. Xception варто відати перевагу, якщо є доступ до потужних обчислювальних ресурсів, таких як GPU або TPU, і немає строгих обмежень на час інференції або розмір моделі; пріоритетом є максимально можлива точність і ви працюєте зі складними наборами даних зображень, де модель повинна отримувати та обробляти велику кількість ознак; є великий обсяг даних для навчання, що знижує ризик перенавчання; існують завдання трансферного навчання, для яких була попередньо навчена модель Xception (наприклад, ImageNet). MobileNetV3 краще використовувати, якщо працюємо на пристроях з обмеженими обчислювальними ресурсами, такими як мобільні телефони або системи, що вбудовуються; для програми критичний час виконання, наприклад, для реального часу або програм з низькою затримкою; якщо програма має бути енергоефективною, як у випадку з батарейними пристроями або довготривалою автономною роботою; дані обмежені, менша ємність мережі MobileNetV3 допоможе уникнути перенавчання; завдання обробки виконуються на пристрої Edge Computing, де важливі як швидкість, так і ефективність.

Таким чином, якщо мета – максимізувати продуктивність та точність на шкоду ресурсам, Xception може бути найкращим вибором. У контексті мобільних пристроїв або коли потрібні більш швидкі швидкості інференції з невеликим споживанням пам'яті та енергії MobileNetV3 є кращим варіантом.

3.5 Техніко-економічне обгрунтування прийнятих рішень

Для апаратної реалізації сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості можна використовувати компоненти, що містять інтегровані елементи штучного

інтелекту. В роботі запропоновано застосування Coral System-on-Module (SoM) [43] – рис. 3.22.

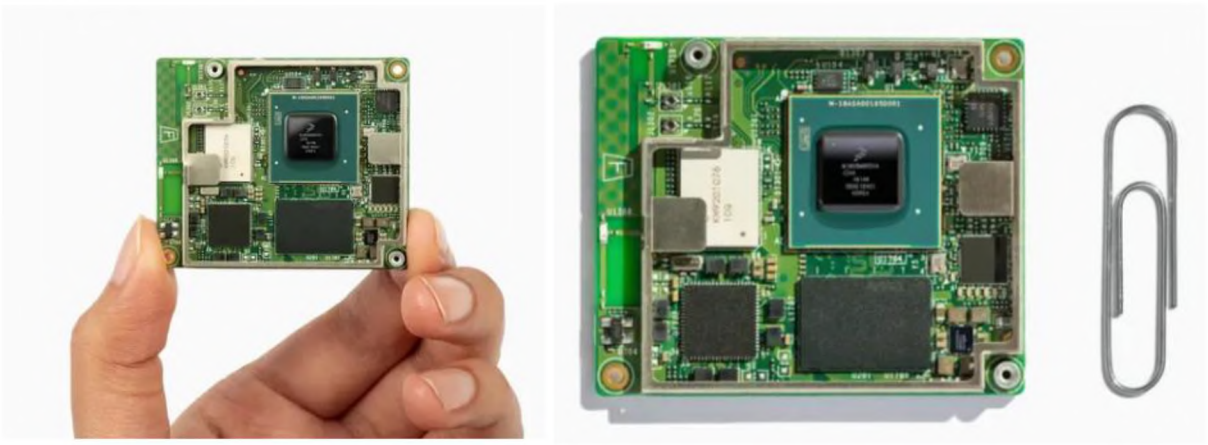

Рисунок 3.22 – Coral SoM

Ця система є комплексною інтеграцією, яка полегшує розробку вбудованих систем, необхідних для ефективного застосування машинного навчання (ML). Вона включає в себе SoC і MX 8M від NXP, флеш-пам'ять eMMC, RAM LPDDR4, а також Wi-Fi та Bluetooth. Особлива відмінність полягає у використанні співпроцесора Edge TPU від Google [44]. Цей компактний ASIC, створений Google, оптимізовано для підсилення TensorFlow Lite моделей. Він забезпечує вражаючу продуктивність – 4 трильйони операцій на секунду (4 TOPS), споживаючи лише 2 Вт енергії, тобто 2 TOPS на ват. Edge TPU може швидко обробляти передові моделі машинного зору, наприклад, MobileNetv2, зі швидкістю до майже 400 кадрів/с. Така можливість обробки даних ML на пристрої значно зменшує затримку, підвищує безпеку даних та усуває необхідність в постійному підключенні до Інтернету. Блок-схема компонентів Coral SoM наведена на рис. 3.23. До переваг даного пристрою варто віднести швидку та енергоефективну обробку машинного навчання з інференцією до 4 TOPS при споживанні енергії всього 2 Вт; ОС Linux («Mendel» – похідна від ОС Debian); блок обробки відео (VPU) дозволяє обробляти дані 4K зі швідкістю 60 кадрів/с. Співпроцесор Edge TPU – мікросхема ASIC, розроблений Google,

яка забезпечує високоефективне логічний вивід для моделей TensorFlow Lite [45], використовує PCIe Gen2 x1 і I_2C/GPIO.

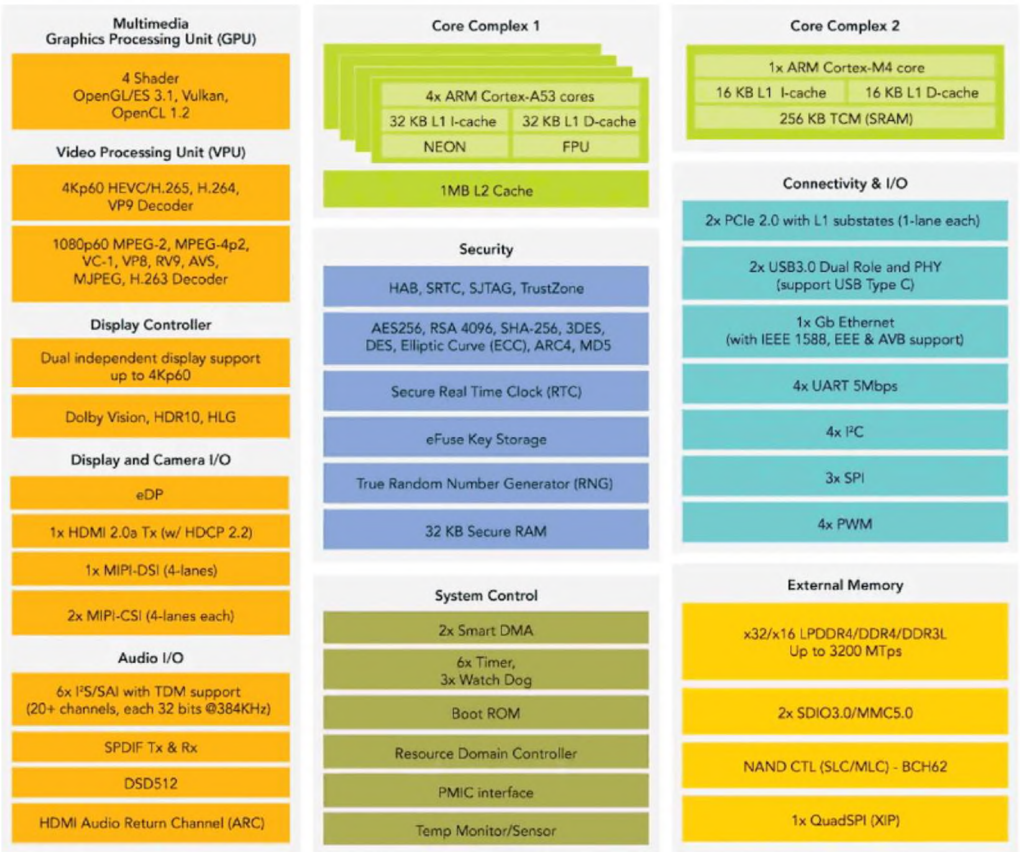

Рисунок 3.23 – Блок схема Coral SoM

Враховуючі необхідність реалізації проєктів Edge Computing + AI CV, плата містить 2 контролери USB 3.0, 1 інтерфейс uSDHC, по 2 модуля UART, I_2C, SPI, 16 ліній GPIO, 4 лінії ШІМ, 2 входи камери MIPI-CSI2, Модулі Wi-Fi (MIMO 2×2), Bluetooth і GE. На даний час , вартість такого пристрою складає 7717 грн.

При цьому витрати на реалізацію програмного компоненту сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості з врахуванням застосування конструктору коду TensorFlow Lite складає 48000 грн.

Таким чином, витрати на реалізацію запропонованого проєкту Edge Computing + AI CV, що визначає сезонну специфікацію зображень складають ≈ 56000 грн.

Висновки до розділу 3

Для оцінки моделі глибокого навчання синтезованої нейронної мережі розглянуті програмно-апаратні компоненти, що базуються на хмарному сервісі Google Colaboratory на мові Python.

Для реалізації класифікатору сезону сформований датасет у пропорції 70 % – на навчальну вибірку та 30 % – на перевірочну, що містить зображення двох класів: поверхня землі зі снігом (клас SNOW) – 154 шт. та без нього (клас NOT_SNOW) – 196 шт. Світлини зроблені з борту пасажирського літака.

В якості метрики для оцінки моделей вибрано Balanced Recall. В роботі досліджено дві моделі основі CNN MobileNetV2 і Xception. Перша містить $\approx 2{,}4\times10^6$, а друга – $\approx 21\times10^6$ параметрів.

Найбільш вдалий варіант моделі MobileNetV2 забезпечує продуктивність 96,2 % на перевірочні виборці, а на базі Xception – 97,5 %.

Xception і MobileNet оптимізовані для різних сценаріїв використання. Для вибору моделі глибокого навчання між нейронними мережами Xception і MobileNet проаналізовано базові чинників, що залежить від вимог до завдання.

Для підтвердження можливості практичної реалізації проєкту Edge Computing + AI CV, що містить сезонно-специфічний сегмент багаторівневої архітектури нейронної мережі класифікації зображень місцевості, виконане техніко-економічне обґрунтування прийнятих рішень. Витрати складають $\approx$ 56000 грн.

ВИСНОВКИ

Запропонована структура сезонно-специфічного сегменту складається з двох частин. Такий підхід орієнтований на реалізацію проєктів Edge Computing + AI + CV.

Перший блок призначений для класифікації сезону, що відповідає зображенню або кадру відеопотоку. Для його синтезу запропоновано дві альтернативні архітектури MobileNetV2 і Xception.

Щоб визначити їх пріоритетність, проведено дослідження властивостей цих CNN, та розглянуті особливості реалізації моделі глибокого навчання нейронної мережі класифікації сезону. При цьому основний акцент зроблений на впровадження Transfer Learning.

У другому блоці сезонно-специфічного сегменту, відповідно до сезону, вирішується основне задання. Наприклад, це може бути: реалізація Object Detection, Object Tracking, сегментація, автопілот UAV та ін. В якості базової розглядається YOLOv8, що містить кілька модифікацій моделі, що орієнтовані на різноманітні за вимогами до обчислювальних потужностей обробки у реальному часі додатки, в тому числі і варіанти Edge Computing + AI + CV.

В рамках оцінки моделі глибокого навчання синтезованих нейронних мереж на базі MobileNetV2 і Xception визначені програмно-апаратні компоненти, що базуються на хмарному сервісі Google Colaboratory на мові Python. Запропонований підхід передбачає Transfer Learning та перенесення сформованих моделей глибокого навчання за допомогою TensorFlow Lite на кінцеві пристрої IoT.

Для виконання зазначеної оцінки продуктивності класифікатору сезону сформований датасет у пропорції 70 % – на навчальну вибірку та 30 % – на перевірочну. Він містить зображення двох класів: поверхня землі зі снігом – 154 шт. (клас SNOW – 154) та без снігу – 196 шт. (клас NOT_SNOW). В якості метрики для оцінки моделей вибрано Balanced Recall.

Як наслідок, в роботі проаналізовано дві моделі на основі MobileNetV2 ($\approx 2{,}4\times10^6$ параметрів) і Xception ($\approx 21\times10^6$ параметрів). Найбільш вдалий варіант моделі MobileNetV2 забезпечує продуктивність 96,2 % на перевірочні виборці, а Xception – 97,5 %.

В цілому, Xception і MobileNetV2 оптимізовані для різних сценаріїв використання. Для вибору моделі глибокого навчання між нейронними мережами Xception і MobileNetV2 проаналізовано базові чинників, що залежить від вимог до завдання.

Для підтвердження можливості практичної реалізації проєкту Edge Computing + AI CV, що містить сезонно-специфічний сегмент багаторівневої архітектури нейронної мережі класифікації зображень місцевості, виконане техніко-економічне обґрунтування прийнятих рішень. Витрати складають $\approx$ 56000 грн.

Таким чином, результатами роботи є структура сезонно-специфічного сегменту багаторівневої архітектури нейронної мережі класифікації зображень місцевості; моделі глибокого навчання згорткових нейронних мереж класифікації сезону зображень; рекомендації щодо практичної реалізації проєкту Edge Computing + AI CV що містить зазначений сезонно-специфічний сегмент. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні хмарних сервісів.