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: **«Вплив дисбалансу датасету на точність класифікації нейронної мережі»**

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_22
Раскін О. М.
Керівник: Слюсар В. І.
Рецензент: Муравльов В. В.

Полтава – 2023 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю використання якісної бази зображень для створення продуктивної моделі глибокого навчання нейронної мережі класифікації зображень. У процесі формування датасету необхідно враховувати роздільну здатність зображень; формат зображень; колірний простір; аугментація; баланс класів; виключення дублікатів; консистентність міток; відділення фону; сценічні умови; пропорційний поділ; метадані. Однак питання впливу дисбалансу датасету на продуктивність моделі глибокого навчання 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 та розпізнавання об'єктів на зображенні за допомогою нейронної мережі, що інтегрується в архітектуру хмарного сервісу.

Завданнями кваліфікаційної роботи є:

- обґрунтування вимог до датасету;
- вибір архітектури нейронної мережі для класифікації зображень;
- реалізація Transfer Learning;

– оцінка впливу дисбалансу на продуктівність нейронної мережі;

– обґрунтування рекомендацій щодо використання моделі глибокого навчання згорткових нейронних мереж класифікації зображень.

Об'єктом дослідження є процес класифікації зображень.

Предметом дослідження є точність нейронної мережі, що застосовуються для класифікації зображень.

Методами дослідження є аналітичний, інформаційно-пошуковий, методи синтезу та навчання нейронних мереж класифікації зображень, робота з фреймворком Keras.

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

Елементи наукової новизни роботи полягають у створенні моделі глибокого навчання згорткових нейронних мереж класифікації зображень.

Практична значущість роботи полягає у розробці рекомендацій щодо використання моделі глибокого навчання згорткових нейронних мереж класифікації зображень – можуть бути використані для подальших досліджень за даною тематикою та при проектуванні хмарних сервісів.

Апробація результатів відбувалася в рамках XXII Міжнародної науково-технічної конференції «Приладобудування: стан і перспективи», (травень 2023 р., м. Київ), V-ої Міжнародної студентської конференції «Цифровізація науки та сучасні тренди її розвитку» (листопад 2023 р., м. Житомир).

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 76 сторінок формату А4. Вона містить 29 рисунків.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ФОРМУВАННЯ ДАТАСЕТУ ДЛЯ ЗАВДАНЬ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Класифікація датасетів

На даний час, штучний інтелект (ШІ) на основі нейронних мереж [1] найчастіше використовується для вирішення різноманітних завдань, наприклад: класифікація, оцінка (регресія), прогнозування часових рядів, сегментація, Object Detection [2], розпізнавання тексту (OCR) [3], розпізнавання мови, генерація зображення (мови), чат-боти та ін.

Нейронній мережі, яке б завдання не вирішували, знадобиться датасет, на якому вона навчатиметься (рис. 1.1). Датасет (від англ. «dataset») – це сукупність даних, зазвичай, структурованих та організованих певним чином [4].

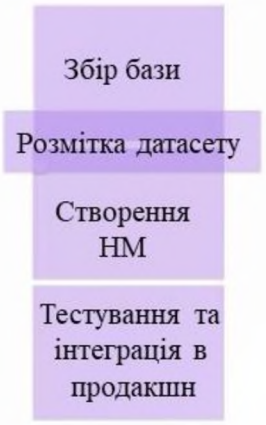

Рисунок 1.1 – Основні складові ШІ на основі нейронних мереж

Він створюється на основі зібраної бази даних або знань. До основних типів даних відносяться: зображення, відео, тексти, табличні дані, часові ряд, аудіо, мова. При цьому абсолютно все рівно який алгоритм навчання використовується. Навіть у разі навчання з підкріпленням незважаючи на те, що на стартовому етапі база взагалі не потрібна, а датасет збирається у процесі функціонування нейронної мережі. Тому від того, які дані будуть

зібрані, багато в чому залежить якість роботи нейронної мережі [5]. На даний час, прийнято вважати, що успішно зібрана база може становити 50-60 % успішності взагалі всього проекту. Важливість створення якісного датасету підтверджується кількома чинниками.

1. Підвищення точності. Розглянемо наступний практичний приклад: за допомогою НМ потрібно визначити зарплату фахівця-нейронника. Середня заробітна плата складає 56000 грн. Було зібрано 5000 резюме. Такий обсяг дає розкид ±22000 грн на перевірочній вибірці, що є досить великим та неприйнятним (більше 1/3). В подальшому використали 50000 резюме. Це дало розкид лише ±6000 грн. Тепер, теоретично, якщо взяти 500000 резюме, то розкид складатиме ±3000 грн (це без процедури очищення викидів, яку потрібно виконувати досить обережно).

2. Оцінка характеристик проєкту, наприклад, пілотного: терміни виконання, вартість, необхідна відеокарта, в якому форматі буде реалізований датасет та ін. Особливо це положення є актуальним, при залученні аутсорсингу.

3. Повне тестування. Іноді виникає ситуація використання невеликого датасету, наприклад: 800 зразків (значень, зображень та ін.) і на тест виділяємо всього 10 % вибірки (80 зразків). В даному випадку може статись нерелевантна оцінка (тобто самі випробування) архітектури нейронної мережі. В ідеалі, треба було б 20 % виділити на перевірочну та 20 % на тестову вибірки. Тобто база у 5000 зразків з розподілом на вибірки: 3000 – навчальна, 1000 – перевірочна та 1000 тестова, вирішила б цю проблему.

Датасети можуть включати різні типи даних, такі як:

1. Табличні дані [6]: записи з одним або декількома атрибутами. Наприклад, база даних клієнтів з їхніми іменами, адресами та історією покупок.

2. Зображення [7]: колекція фотографій або зображень, які можна використовувати для завдань комп'ютерного зору, таких як розпізнавання об'єктів або класифікація зображень.

3. Текст [8]: збірки статей, блогів, відгуків та ін. Ці дані часто використовують у завданнях обробки природної мови.

4. Часові ряди [9]: послідовні дані, зібрані в хронологічному порядку, наприклад, дані про погоду або котирування акцій.

5. Відео та аудіо [10]: записи, які можуть бути проаналізовані для різних цілей, наприклад, для розпізнавання мови або аналізу змісту відео.

6. Інші [11]: також існує безліч інших типів даних, залежно від конкретної галузі дослідження або застосування.

Датасети часто поділяють на підмножини для навчання, валідації та тестування моделей. Це дозволяє оцінити продуктивність моделі на даних, які вона раніше не бачила, і запобігти перенавченню. В цілому, датасети можна класифікувати наступним чином.

1. За варіантами наявності: готові, є невелика частина, взагалі немає.

2. За можливістю нарощування: легко, складно нарощувати, нарощується з часом (продаж компанії, звернення в технічну підтримку, наприклад, за 5 років – 10000 звернень та ін.), не можна наростити (зустрічається досить поодиноко, наприклад: вкрай рідкісні захворювання, вихід з ладу обладнання на АЕС).

Якщо немає готового датасету, то існують такі види його збору: збір з оплатою за одиницю – найчастіший; збір за часом (захворювання, продаж компанії); парсинг (збір в Інтернеті по сайтам); партнерство.

1.2 Аналіз вимог до наборів даних для датасету

Вимоги до наборів даних для створення датасету залежать від конкретного завдання. Однак є низка загальних рекомендацій та кращих практик, які варто враховувати при створенні датасету.

Розмір даних – чим більше даних, то краще, особливо якщо плануєте використовувати глибоке навчання. Однак даних має бути достатньо, щоб

вони були репрезентативними для завдання. На сьогоднішній день обсяг бази є визначальним для навчання нейронної мережі [12]. На жаль, НМ поки що не здатна навчитися на маленьких датасетах. Процес розробки та досліджень у цьому напрямі ведуться. Але чогось конкретного поки що не придумали і не винайшли. А тому обсяг бази є визначальним. При цьому не можна відповісти на запитання: який обсяг датасету необхідний чи скільки прикладів потрібно для того, щоб навчити проект? Спочатку неможливо сказати скільки прикладів потрібно для того, щоб навчилася нейронна мережа. Правило тут просте – що більше прикладів буде зібрано, то точніше буде модель. Правильно говорити про таке поняття, як мінімальний обсяг. Тобто певна кількість прикладів, які необхідні для того, щоб почати тестування моделей, що розробляються. Знову ж таки багато що залежить від того завдання, яке має нейронна мережа. Наприклад, для завдання класифікації зображень часто буває досить 300 прикладів кожного класу; Для задачі сегментації зображень – 1000 зображень кожного класу, в принципі, також може бути достатньо. Це мінімальний поріг, який потрібно зібрати для того, щоб почати будувати свої архітектури. Паралельно не забуваючи, що датасет потрібно розширювати.

Якість даних – дані мають бути чистими та без помилок (помилкові або спотворені дані можуть призвести до поганих результатів під час навчання моделей) [13]. Цьому аспекту також слід приділяти увагу. Особливо якщо база збирається не самим розробником нейронної мережі. Залежно від того, де формувався датасет, необхідно перевірити, наскільки актуальні та вірні дані були зібрані. Наприклад, для нейронної мережі класифікації автомобілів замовлений фрілансерам датасет із 30 марок автомобілів, серед яких зазначені Tesla, Jaguar тощо (рис. 1.2). При цьому в теці з автомобілями Jaguar можуть бути зображення тваринного ягуара. Або в папці з автомобілями Tesla знаходяться портрети Миколи Тесла. Звичайно, все це позначається на результаті роботи нейронної мережі. Як наслідок, треба виконувати фільтрацію даних.

Балансування – залежно від завдання може знадобитися балансування класів (наприклад, у завданнях класифікації незбалансовані датасети можуть призвести до поганої якості моделі). Необхідно звертати увагу на класи, тобто вони мають бути збалансовані. На рис. 1.3 наведений незбалансований датасет.

Рисунок 1.2 – Помилкові зображення при зборі датасету

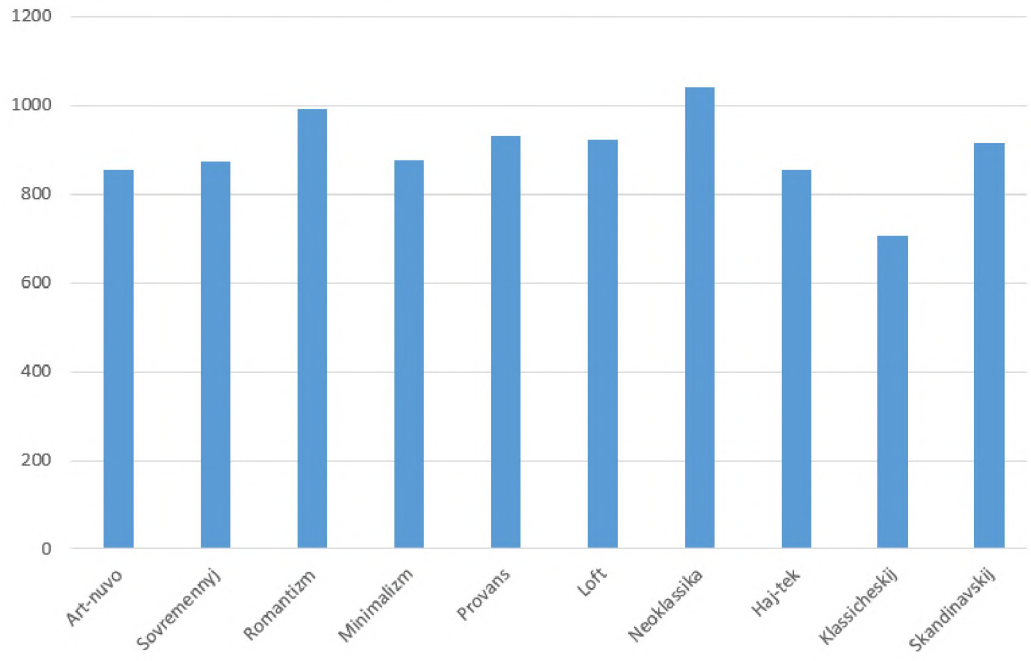

Рисунок 1.3 – Приклад незбалансованого малого датасету

Варіативність – дані мають бути різноманітними та покривати всі можливі сценарії та класи, які ви хочете врахувати. Для цього потрібно виконувати збагачення датасету. Чим більше зображень і різноманітнішими вони будуть у датасеті, тим більш точну модель глибоко навчання можна навчити. Тут перетинаються пункти, пов'язані із балансуванням. Але це трохи інше. Наприклад, якщо стоїть завдання написати нейронну мережу, яка розрізнятиме марку Nissan від марки Mitsubishi. При цьому зібрано базу, яка містить 10000 зображень з Nissan і 10000 зображень з Mitsubishi. Але при цьому виявилося, що в папці з автомобілями Nissan присутні лише 3 моделі, а в папці з Mitsubishi – тільки одна. Враховуючи обсяг датасету, потенційно можна реалізувати досить потужну нейронну мережу, яка дає якісні результати, наприклад, на тестовій вибірці точність – 99 %. Але, згодом, якщо тестувати модель зображень, наприклад, Nissan Juke, то з певною часткою ймовірності, нейронна мережа зможе визначити, що ця машина відноситься до марки Nissan. Але далеко не факт, тому що нейронна мережа жодної машини у такій комплектації не бачила. Тобто складно припустити, як вона спрацює у реальних умовах на цій машині. Те саме стосується Mitsubishi. Якщо візьмемо якусь нову модель, то передбачити якість роботи нейронної мережі буде складно. Бажано, щоб датасет містив усі можливі варіанти. При цьому може бути ситуація, коли додали до бази варіативності, але кількість нових прикладів все одно виявилася недостатньою. Потрібно не просто збільшити датасет, а ще зробити це якісно та грамотно, не втративши в пункті варіативності.

Подання даних – датасет зберігається у форматі, який підходить для роботи з обраною бібліотекою машинного навчання, наприклад, можна зберегти зображення у форматі JPEG або PNG, а позначки класів у форматі CSV або JSON. Дані мають бути у зручному форматі для обробки та аналізу (це може включати нормалізацію, масштабування або інші перетворення).

Розподіл на піднабори [14]. При створенні датасету отримана база розподіляється на кілька виборок: навчальна, перевірочна (валідаційна),

тестова (рис. 1.4). Навчальна вибірка використовується для навчання мережі, а валідаційна вибірка – використовується у процесі навчання з метою оцінки якості навчання. По суті, під час пошуку оптимальної архітектури намагаємось отримати на перевірочній вибірці гарний результат у точності.

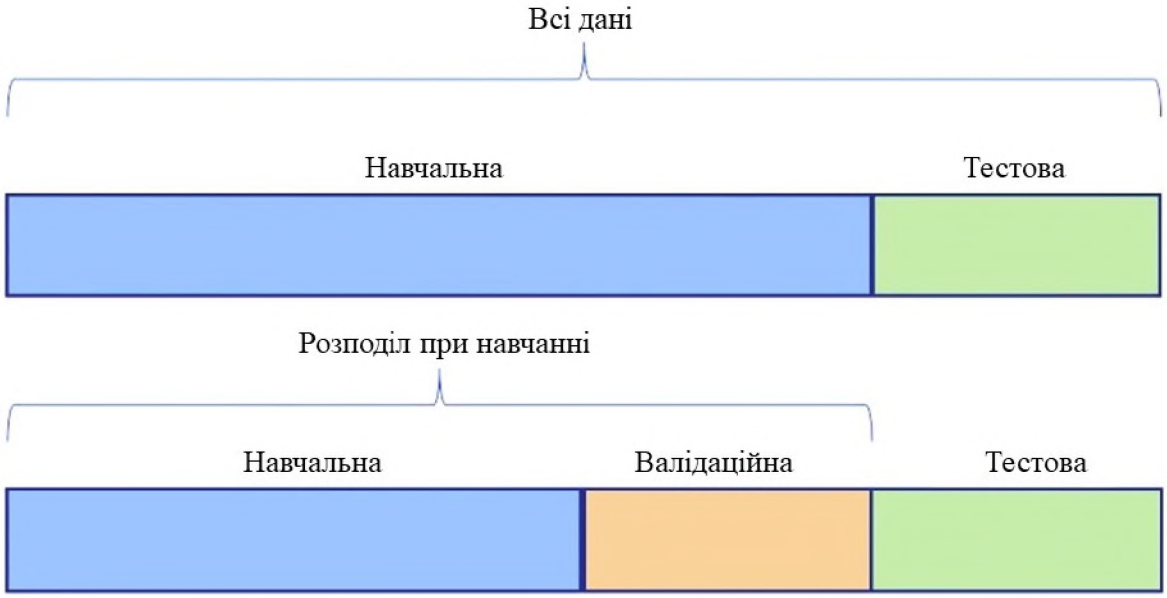

Рисунок 1.4 – Типові складові частини датасету

Перевірочна вибірка необхідна для ситуацій, коли на навчальній вибірці нейронна мережа показує 100 % точність, але за фактом погано працює. Це означає, що мережа вивчила всі приклади і тепер не передбачає, а видає завчені відповіді. По суті, використання валідаційної вибірки – перший крок до оцінки якості роботи моделі глибокого навчання. Цей набір використовується на кожній епосі навчання. Важливо, щоб дані з перевірочної та навчальної вибірок не перетиналися та не повторювалися. Якщо модель видає на цій вибірці більше 95% можна вважати, що мережа чудово працює. Однак і в цьому випадку, існує імовірність підлаштовування під дані що входять до складу датасету.

Тестова вибірка використовується для оцінки якості роботи мережі після завершення навчання, тобто для підготовки моделі в Production [15]. Зазвичай, вона зберігається у замовника та не надається розробникам. Іноді,

вона є у окремого тимліда (керівник проекту). Наприклад, якщо продуктивність нейронної мережі складає на навчальній 99%, на перевірочній 97%, а на тестовій 92% виборках, то свідчить про підлаштовування під датасет. Якщо з'являються нові дані, бажано проводити оновлення датасету. Особливо це актуально для часових рядів, наприклад вартість квартири.

Найчастіше зустрічається пропорція розподілу датасету така: 80% – 15% – 5% або 70% – 20% – 10% [16]. Але, залежно від завдання, він може ділити по іншому. Таким чином, у межах навчання рідко застосовується тестова вибірка, і датасет ділитися лише дві. У реальних завданнях краще залишати дані під тестову вибірку, щоб бути впевненим у результатах.

Якщо вирішується завдання класифікації (наприклад, зображень), то надалі необхідно перемішати дані, оскільки класи розташовані послідовно. Як і нормалізація, перемішування необхідне для рівномірного навчання нейронної мережі. Причому дані масиву зображень, так і масиву міток повинні бути перемішані узгоджено. Тільки так збережеться взаємно однозначна відповідність між масивом фотографії та його міткою класу [17]. Анотація та мітки – для навчального датасету потрібні правильні (послідовними та точними) мітки або анотації.

Уникання витоку даних – треба переконатись, що інформація з тестового набору не просочилася до навчального набору [18].

Документація – важливо надати документацію або опис створеного датасету. Це допоможе іншим членам команди розробників зрозуміти його зміст, походження та будь-які особливості [19].

Конфіденційність – дані не містять чутливої інформації або вони анонімізовані з дотриманням законодавчих норм щодо захисту даних.

Крім цих основних пунктів, можуть бути специфічні вимоги в залежності від сфери застосування, типу даних або конкретної задачі [20].

1.3 Розмітка датасету

Розмітка датасета є ключовим етапом для навчання з учителем. Особливо це стосується обробки зображень. Для того, щоб нейронна мережа вміла розпізнавати об'єкти та образи на зображенні, потрібно їй спершу «показати», що саме там знаходиться. Для цього і призначено розмітку. Розмітка зображень (Image Annotation, анотування зображень, анотація зображень) є невід'ємною частиною розробки ШІ, і це одне з основних завдань технології комп'ютерного зору [21]. Анотовані зображення потрібні як вступні дані для навчання нейронних мереж: розпізнавання об'єктів на зображеннях дозволяє комп'ютерам «бачити» навколишній світ подібно до людини. Розглянемо декілька прикладів. На рис. 1.5 наведена розмітка для завдання Object Detection (розмітка прямокутниками).

Рисунок 1.5 – Розмітка для Object Detection

Для завдань сегментація потрібно обвести об’єкт по контуру за допомогою Photoshop або спеціалізованим ПЗ зробити маску – рис. 1.6.

Однак при великій кількості класів сегментування ціна за обробку зображення може значно зрости. На щастя, для сегментації потрібен не так вже й великий датасет – близько 1000 картинок достатньо. У випадку обробки мультикласових даних виконується тегування (рис. 1.7). Є ще завдання Pose Estimation – для людини це визначення пози (майже як Object Detection тільки розмічуються ключові точки) – рис. 1.8 [22, 23].

Рисунок 1.6 – Розмітка для сегментації

Рисунок 1.7 – Мультикласові дані

Рисунок 1.8 – Приклад Human Pose Estimation

В цілому, на основі проведеного аналізу слід виділити кілька типових випадків, коли необхідно проводити розмітку датасету.

1. Навчання з вчителем – якщо використовуються методи супервізованого навчання, тонеобхідно мати мітки для кожного зображення у датасеті.

2. Перенавчання моделі – існує необхідність доопрацювання (finetune) попередньої навченої моделі під специфічні дані.

3. Створення нового датасету – якщо є унікальні дані, які не представлені в існуючих датасетах (розмітка цих даних дозволить створити новий датасет для навчання або тестування моделей).

4. Оцінка продуктивності моделі.

5. Багатокласова або багатоміткова класифікація – якщо зображення можуть належати до кількох класів одночасно або якщо є множина потенційних класів, то потрібно буде розмітити кожне зображення відповідними мітками.

6. Детектування дефектів або аномалій на зображеннях (коли норма та аномалії чітко визначені) знадобиться розмічений датасет для навчання та валідації моделі.

7. Семантична сегментація. Хоча це не класичне завдання класифікації зображень, але для семантичної сегментації, де кожен піксель зображення класифікується до певного класу, також потрібна розмітка.

8. Навчання з частковим учителем та активне навчання. У цих методах початкові розмічені дані використовуються для навчання базової моделі, яка потім може бути доповнена з використанням нерозмічених даних.

9. Поліпшення якості існуючого датасету – якщо є імовірність вважати, що існуючі мітки у датасеті є ненадійними або неточними, то можна провести повторну розмітку.

10. Створення навчальних даних для моделей, що працюють на рівні примірника – якщо завдання вимагає класифікації або визначення конкретних

об'єктів на зображенні (наприклад, визначення конкретних типів тварин на фотографії та ін.).

Важливо, що процес розмітки може бути трудомістким та досить затратним завданням, особливо якщо потрібно розмітити великі набори даних. В залежності від складності завдання та кількості даних може знадобитися багато часу та ресурсів. Проте правильна і точна розмітка критично важлива досягнення хорошої продуктивності моделі. Як наслідок, може виконуватись автоматична анотація зображень (Automatic Image Annotation, також відома як автоматичне маркування зображень). Це процес, у якому комп'ютер автоматично надає метадані цифрового зображення (підписи або мітки), використовуючи відповідні ключові слова для опису його візуального змісту. Існуючі алгоритми розмітки можна розділити на дві категорії: методи навчання на основі моделей – вони досліджують кореляцію між візуальними характеристиками та їх семантичним значенням для виявлення функції відображення за допомогою машинного навчання або моделей представлення знань для розмітки зображень; моделі на основі бази даних – вони одразу видають послідовність ймовірних міток (лейблів) відповідно до вже анотованих зображень у базі даних.

1.4 Обґрунтування специфічних вимог до датасету для моделі глибокого навчання класифікації зображень

Для створення датасету, призначеного для класифікації зображень, необхідно враховувати низку специфічних вимог [24].

Роздільна здатність зображень. Залежно від вашого завдання та моделі, визначте відповідну роздільну здатність зображень. Занадто висока роздільна здатність може призвести до великих обчислювальних потужностей, в той час як занадто низька може призвести до втрати важливих деталей.

Формат зображень. Зазвичай, використовуються формати JPG або PNG, але вибір залежить від вимог до якості та інших параметрів.

Колірний простір. Всі зображення повинні мати однаковий колірний простір (наприклад, RGB).

Аугментація. Для збільшення різноманітності та покращення навчання моделі можна використовувати аугментацію зображень (повороти, масштабування, обрізання, зміна яскравості тощо).

Баланс класів. Дуже важливо мати приблизно однакову кількість зображень для кожного класу, щоб модель не стала зміщеною у бік представленого класу.

Виключення дублікатів. Треба переконатися, що у датасеті відсутні дублікати або дуже схожі зображень, які можуть спотворити результати.

Консистентність міток. Мітки мають бути послідовними та точними. Найкраще, якщо інструкція виконується однією групою осіб або з використанням чітко визначених інструкцій.

Відділення фону. У деяких завданнях класифікації може знадобитися, щоб об'єкти на зображеннях було чітко видно на нейтральному або однаковому фоні.

Сценічні умови. Потрібно звертати увагу на умови освітлення, кути зйомки та інші аспекти, які можуть впливати на зовнішній вигляд об'єктів на зображенні.

Пропорційний поділ. При поділі на навчальний, валідаційний та тестовий набори треба переконайтеся, що розподіл класів приблизно однаковий у всіх піднаборах.

Метадані. Увімкніть метадані, якщо вони доступні та релевантні (наприклад, місце та час зйомки, використовуване обладнання та ін.).

Дотримання цих вимог допоможе створити якісний та репрезентативний датасет для класифікації зображень. В свою чергу, щоб визначити кількість зображень одного класу необхідно для глибокого

навчання нейронної мережі класифікації зображень необхідно враховувати низку вагомих чинників:

– перенесення навчання (Transfer Learning) – якщо використовується заздалегідь навчена модель, то може знадобитися набагато менше зображень. Попередньо навчені моделі, такі як VGG, ResNet та ін., навчалися на мільйонах зображень, і їх можна донавчити на конкретній задачі з набагато меншою кількістю даних [25];

– якість даних – добре розмічені, чисті та різноманітні дані можуть зменшити необхідну кількість навчальних зображень;

– складність завдання – для завдання, де зображення добре помітні (наприклад, коти проти вантажівок), потрібно менше зображень, ніж для завдання, де відмінності менш очевидні (наприклад, породи кішок);

– складність моделі – великі та складні моделі, як правило, вимагають більше даних для навчання, щоб уникнути перенавчання;

– Data Augmentation – методи збільшення даних, такі як повороти, масштабування, відображення та зміна яскравості можуть помножити кількість ваших даних, роблячи модель більш стійкою до різних варіацій на зображенні [26];

– різноманітність класів – якщо є множина класів і різниця між ними тонка, ймовірно, знадобиться більше зображень на клас.

В цілому, для простих завдань може знадобитися кілька сотень зображень на клас (традиційно вважається приблизно одна тисяча), для складніших завдань – тисячі або навіть десятки тисяч. На практиці рекомендується почати з того, що є, і поступово додавати дані, спостерігаючи за поведінкою моделі у процесі навчання та валідації.

Збільшення обсягу датасету для класифікації зображень часто використовується для підвищення продуктивності моделі, особливо коли спочатку доступна невелика кількість даних. Існує кілька методів для розширення датасету:

1. Аугментація даних (Data Augmentation) [27]:

– геометричні трансформації – повороти, зрушення, масштабування, відбиття;

– колірні трансформації – зміна яскравості, розмаїття, насиченості;

– шум – додавання випадкового шуму на зображення;

– розмиття – застосування різних видів розмиття;

– випадкові обрізки (random crops) – обрізання різних частин зображення.

– еластичні деформації – особливо корисні для медичних зображень.

2. Синтез даних (Data Synthesis) :

– генерація зображень за допомогою GAN (Generative Adversarial Networks) – GAN можуть бути навчені створювати зображення, які схожі на ті, що вже є у вашій датасеті;

– змішування зображень: комбінування різних зображень для створення нового.

– 3D-моделювання – створення та рендеринг 3D-моделей для отримання зображень з різних ракурсів.

3. Зовнішні джерела даних:

– публічні датасети – інтеграція з іншими існуючими датасетами, які мають ті ж чи схожі класи.

– веб-скрейпінг – автоматичне вилучення зображень з Інтернет (з урахуванням авторських прав та інших юридичних аспектів).

4. Перенавчання на інших датасетах (Transfer Learning). Хоча це технічно не збільшує ваш датасет, перенавчання попередньо навченої моделі на вашому невеликому датасеті може допомогти досягти кращої продуктивності, якби у вас було більше даних.

5. Активне навчання: почати з меншого підмножини розмічених даних, навчити модель, а потім використовувати її для передбачення міток не розмічених даних. Потім експерти можуть перевіряти та коригувати ці позначки, і процес може повторюватися.

6. Кооперація та партнерство: співробітництво з університетами, дослідницькими інститутами або іншими організаціями для об'єднання даних або спільної розмітки.

При цьому, веб-скрейпінг часто використовується для збору даних з Інтернет, коли немає доступу до API або іншого зручного способу завантаження даних. Для даного методу можна використовувати існуючі бібліотеки (наприклад, у Python [28] популярними інструментами є Beautiful Soup та Scrapy), а також веб-драйвери, наприклад, Selenium – це інструмент, який дозволяє керувати браузером для автоматизації дій на веб-сторінках, що може бути корисним для сайтів із динамічним завантаженням контенту. Спочатку надсилається запит HTTP на цільову сторінку. Після отримання відповіді вміст сторінки (зазвичай HTML) аналізується для отримання потрібних даних. За допомогою селекторів (наприклад, CSS-селекторів або XPath) із вмісту сторінки виймаються конкретні дані. Вилучені дані зберігаються в потрібному форматі (наприклад, CSV, JSON або базі даних). Багато сучасних веб-сайтів завантажують контент динамічно за допомогою JavaScript. У разі простого запиту до HTML може бути недостатньо. Інструменти типу Selenium дозволяють «оживляти» сторінки і чекати завантаження динамічного контенту. Не всі сайти дозволяють скрейпінг. Щоб визначити, чи можна скрейпити сайт, варто перевірити файл robots.txt на даному веб-сайті. Скрейпінг може завантажувати сервери веб-сайту. Необхідно бути уважним і не надсилати надто багато запитів за короткий проміжок часу. Однак, для веб-скрепінгу існують певні обмеження: сайти можуть блокувати скрейпери за IP-адресою або іншими ознаками; Деякі сайти використовують різні заходи для запобігання скрейпінгу, такі як CAPTCHA, а також структура веб-сайтів може змінюватись, що може «ламати» використовуваний скрейпер. Як наслідок, потрібно планувати скрейпінг, щоб мінімізувати навантаження на цільовий сайт; використовувати затримки між запитами та розглянути можливість використання проксі-

серверів, щоб уникнути блокування; регулярно перевіряти та оновлювати скрейпер, особливо якщо він критично важливий для проекту чи бізнесу.

З розглянутих підходів, досить прийнятним для збільшення обсягу датасету моделі глибокого навчання класифікації зображень є метод синтезу даних Data Synthesis. Це процес створення нових даних на основі існуючих, але не простою їхньою зміною, як у випадку з аугментацією, а шляхом генерації нових прикладів, які можуть доповнювати або розширювати початковий датасет. Синтез даних часто використовується в областях, де збирання оригінальних даних складне, високовартісне або неможливе. Для класифікації зображень існує кілька методів синтезу даних.

GAN. Така нейронна мережа складається з двох частин – генератора, що створює зображення, та дискримінатора, який намагається відрізнити справжні зображення від згенерованих. Як тільки GAN навчається на Вашому датасеті, генератор може створювати нові зображення, які схожі на ті, які він бачив під час навчання, але є унікальними.

Змішування зображень (Image Blending) – метод полягає в комбінуванні елементів різних зображень для створення нового. Наприклад, голова однієї тварини може бути поміщена на тіло іншої. Хоча це може звучати абсурдно, в деяких контекстах це може бути корисним, особливо якщо комбінування виконується інтелектуально і створює реалістичні зображення

3D-моделювання та рендеринг. Деякі компанії та дослідники використовують 3D-моделювання для створення штучних зображень. Наприклад, для створення зображень об'єктів з різних ракурсів або різних умов освітлення. Цей метод може бути особливо корисним у областях, де реальні об'єкти складно або дорого фотографувати.

Синтез з використанням нейронно-мережних стилів (Neural Style Transfer) дозволяє застосовувати стиль одного зображення до іншого. Це може бути використане для створення нових зображень, які поєднують зміст одного зображення та стиль іншого.

Векторні перетворення у прихованому просторі. Використовуючи заздалегідь навчені моделі, можна перетворити зображення на їх приховане подання (наприклад, вектори), виконати перетворення в цьому просторі та потім інвертувати перетворення, щоб отримати нові зображення.

Синтез даних має свої переваги, тому що дозволяє генерувати велику кількість унікальних зображень, але при цьому необхідно ретельно контролювати якість цих зображень. Однак, не всі згенеровані зображення будуть реалістичні або релевантні для конкретної задачі класифікації.

При використанні цих методів важливо стежити за якістю нових даних, щоб вони були релевантними та не вносили спотворень у навчання моделі. Наприклад, надмірне використання аугментації може призвести до того, що модель надто оптимізована для аугментованих даних і покаже погану продуктивність на реальних даних.

Висновки до розділу 1

Нейронній мережі, яке б завдання не вирішувала, знадобиться датасет, на якому вона навчатиметься. Він створюється на основі зібраної бази даних або знань. Застосування в процесі навчання якісного датасету дозволяє підвищення точності, оцінити характеристики створюваного проєкту; провести повноцінне тестування.

Датасети можна класифікувати за кількома ознаками: за варіантами наявності та за можливістю нарощування. У випадку відсутності готового датасету, його можна зібрати за кількома варіантами, в тому числі залучаючи аутсорсинг, парсинг та партнерство.

Існує загальні вимоги, які варто враховувати при створенні датасету. До них слід віднести: розмір даних; якість даних; балансування; варіативність; подання даних; розподіл на піднабори; анотація та мітки; уникання витоку даних; документація; конфіденційність.

Для того, щоб нейронна мережа вміла розпізнавати об'єкти та образи на зображенні, потрібно виконувати Image Annotation. Анотовані зображення потрібні як вхідні дані для навчання нейронних мереж. При цьому, розпізнавання об'єктів на зображеннях дозволяє комп'ютерам «бачити» навколишній світ подібно до людини. Правильна і точна розмітка критично важлива досягнення хорошої продуктивності моделі глибокого навчання.

При створенні датасету для класифікації зображень необхідно враховувати низку специфічних вимог, які стосуються таких аспектів, як: роздільна здатність зображень; формат зображень; колірний простір; аугментація; баланс класів; виключення дублікатів; консистентність міток; відділення фону; сценічні умови; пропорційний поділ; метадані.

Для простих завдань може знадобитися кілька сотень зображень на клас (традиційно вважається приблизно одна тисяча), для складних завдань – тисячі або навіть десятки тисяч.

Для підвищення продуктивності класифікації зображень доцільно збільшувати обсяг датасету. Серед існуючих методів слід досить часто використовується аугментація.

РОЗДІЛ 2
МОДЕЛЬ ГЛИБОКОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

2.1 Формування датасету для дослідження впливу дисбалансу на точність класифікації нейронної мережі

Для визначення залежності продуктивності моделі глибокого навчання нейронної мережі класифікації зображень в умовах дисбалансу датасету необхідно сформувати набір, я в якому можна регулювати збалансованість класів. В якості базового розглядається набір даних для машинного навчання CIFAR-10 (рис. 2.1) [29].

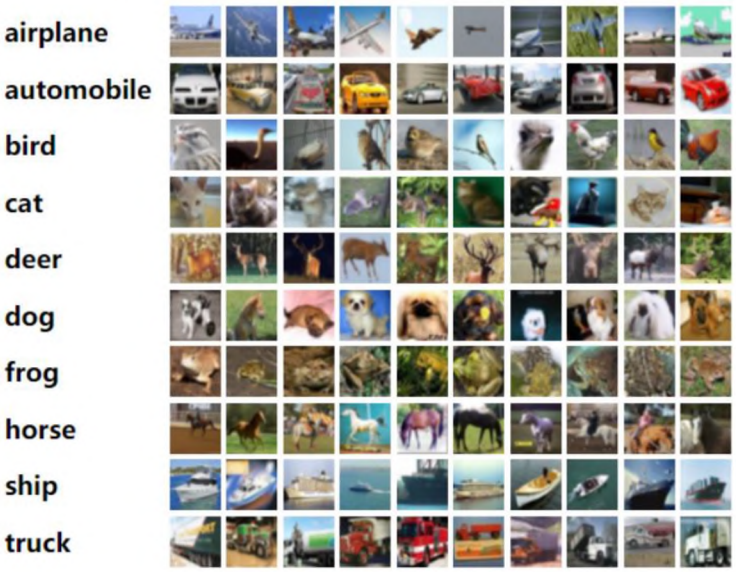

Рисунок 2.1 – CIFAR-10

Він складається з 60000 кольорових зображень розміром 32×32 пікселя, розділених на 10 класів:

1. Літаки (Airplane).
2. Автомобілі (Automobile).
3. Птахи (Bird).
4. Кішки (Cat).
5. Олені (Deer).

6. Собаки (Dog).
7. Жаби (Frog).
8. Коні (Horse).
9. Кораблі (Ship).
10. Вантажівки (Truck).

CIFAR-10, зазвичай, використовується для навчання та тестування алгоритмів машинного навчання, зокрема алгоритмів комп'ютерного зору. Кожен клас представлений однаковою кількістю зображень. Набір даних поділено на 50000 зображень для навчання та 10000 зображень для тестування. В роботі запропоновано створення кількох датасетів, для яких використано два параметри (рис. 2.2):

– $D1$ – максимальний дисбаланс класу відносно початкового розміру вибірки (%) – «амплітуда»;

– $D2$ – дисбаланс між класами (%).

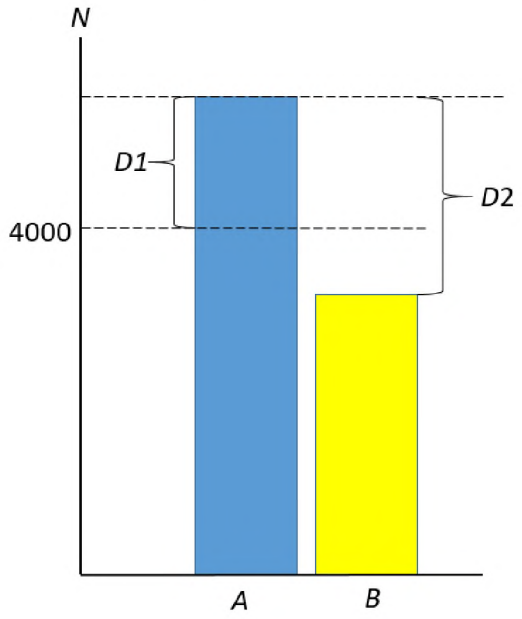

Рисунок 2.2 – Пояснення параметрів введеного дисбалансу $D1$ і $D2$

При цьому, клас A має максимальне відхилення від початкової кількості (4000 шт.) зображень, клас B має мінімальну кількість зображень. Щоб реалізувати певний рівень дисбалансу в кожному класі змінюється кількість зображень, наприклад, при зменшені кількості – у класах фотографії

віднімаються в кінці нумерованого списку фотографій, а при збільшені – додаються на початку нумерованого списку. Щоб усунити витікання даних початкова навчальна вибірка формується з 40000 зображень CIFAR-10 (відповідно у кожному класі – 4000 шт.). Для забезпечення адекватності синтезованої моделі глибокого навчання дисбаланс формується за певним співвідношенням зі значенням від $D1 = 0$ % до $D1 \approx 65$ %. Даний параметр можна трактувати як амплітуда дисбалансу. Для прикладу, вибране наступне фіксоване співвідношення дисбалансу між 10-ма класами:

1. -0,010781.
2. +0,05072.
3. -0,02141.
4. -0,10113.
5. -0,0123.
6. -0,022929.
7. +0,01807.
8. +0,00972.
9. +0,07653.
10. +0,01351.

Тобто максимальний дисбаланс матиме 4 клас.

З одного боку, запас зображень у 1000 шт. вибрано з міркувань компромісу між релевантністю існуючих оцінок моделей глибокого навчання класифікації зображень на основі різних архітектур нейронних мереж для датасету CIFAR-10 і максимальним значення дисбалансу ($D1 \approx 65$ %).

З іншого боку, такий підхід дозволить забезпечити чистоту експериментів шляхом фіксації даних при різних значеннях $D1$ і $D2$ для різних архітектур мереж, використовуючи зображення з набору CIFAR-10.

Таким чином, цією метою з CIFAR-10 формується два набори:

– C10img4000full – $D1 = 0$ %;

– C10img65 – $D1 = 65{,}725$ %.

Розглянемо більш детально кожен з них.

C10img4000full містить 40000 зображень (4000 зображень на кожен клас). За кожним класом тренувальна вибірка складає 3000 зображень, а тренувальна – 1000 шт. Таким чином, для даного набору $D1 = 0$ % і $D2 = 0$ % (рис. 2.3).

C10img65 (рис. 2.4) реалізує $D1 = 65{,}725$ % і $D2 = 77{,}112$ %. В даному випадку, 4-ий клас містить 1371 зображення, а 9-ий клас – 5990 зображень.

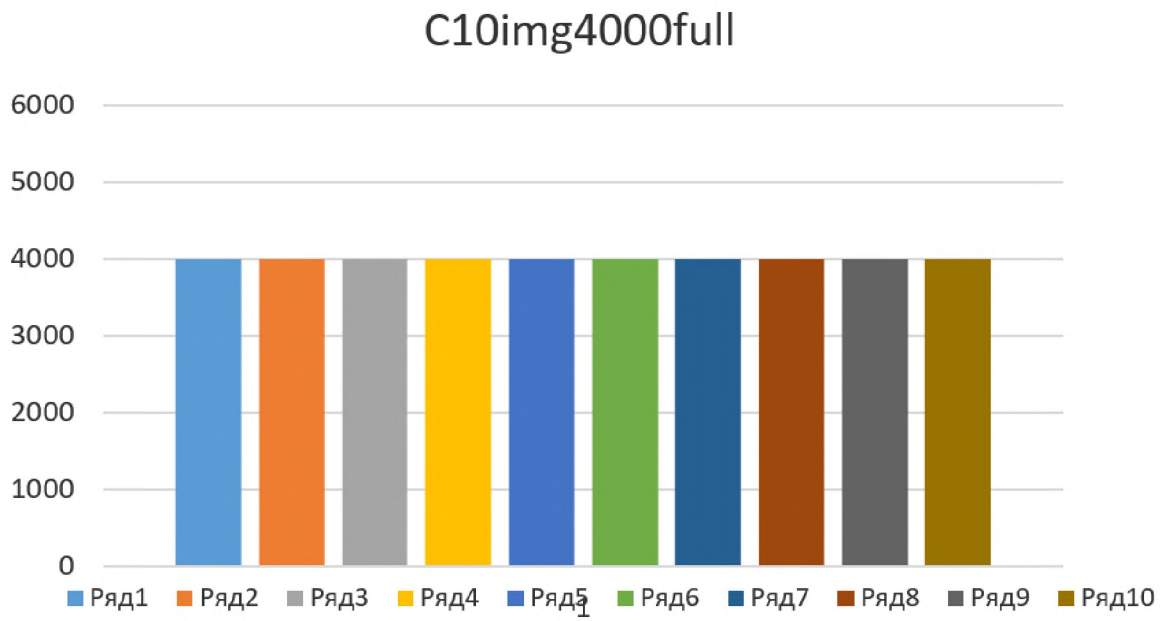

Рисунок 2.3 – Датасет C10img4000full

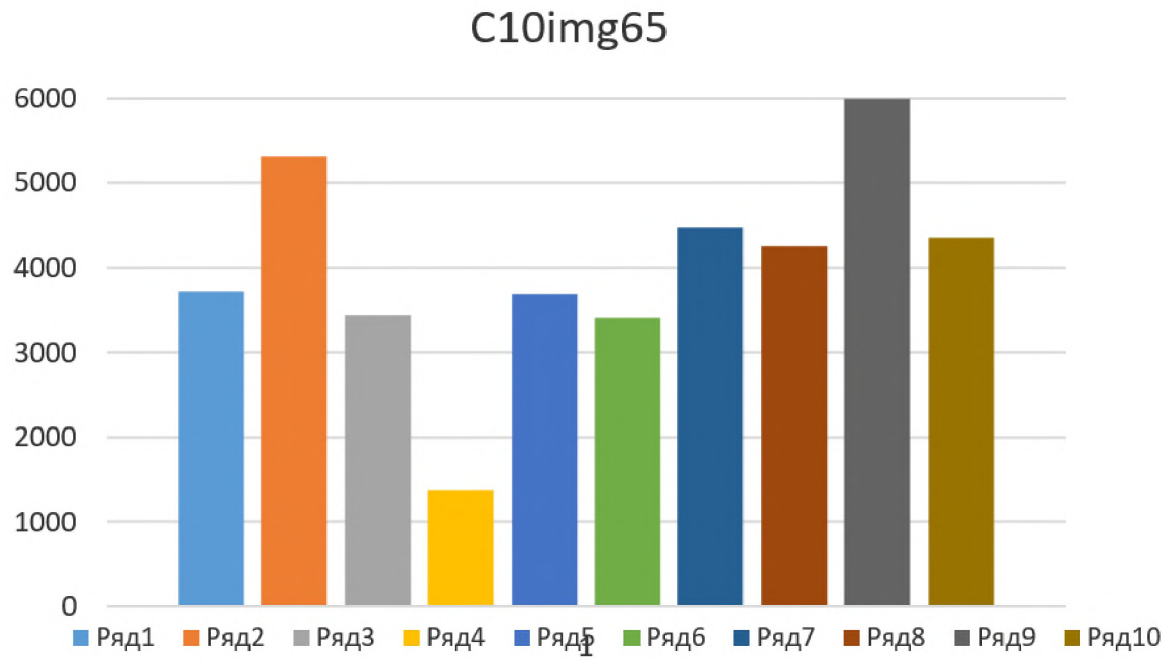

Рисунок 2.4 – Датасет C10img65

2.2 Вибір архітектури нейронної мережі для класифікації зображень

Згорткові нейронні мережі [30] є однією з форм багатошарових нейронних мереж (рис. 2.1). Перша частина складається з шарів згортки і максимального пулу, які виступають як екстрактор ознак. Друга частина складається з повнозв'язного шару [31], який виконує нелінійні перетворення вилучених ознак та діє як класифікатор.

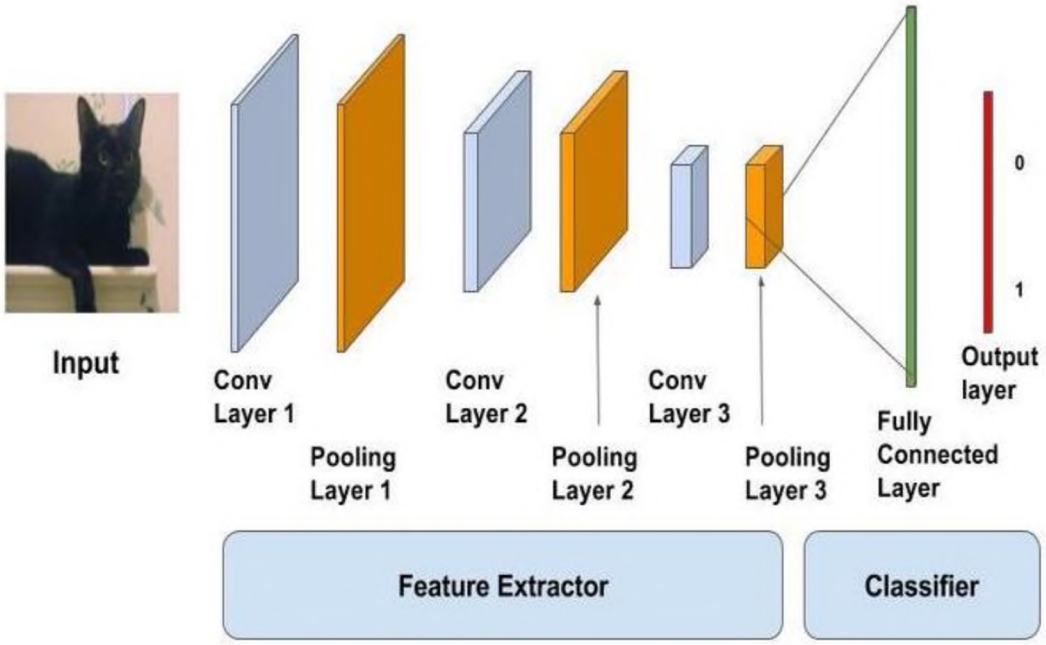

Рисунок 2.1 – Спрощене представлення архітектури CNN

На Рис. 2.2 зібрано багато інформації: тут є інформація про розмір моделі, швидкість, і точність. Тут можна побачити, що VGG – величезні за обсягом моделі (рис. 2.3), але за нинішніми мірками вони мають середню точність. Вони характеризуються появою «стандартних» блоків усередині моделі та згортки 3×3. VGG (Visual Geometry Group) – це архітектура глибокої нейронної мережі, розроблена групою Visual Geometry Group з Оксфордського університету [31]. Ця архітектура стала відома після участі у змаганні з класифікації зображень ImageNet у 2014 р. [32].

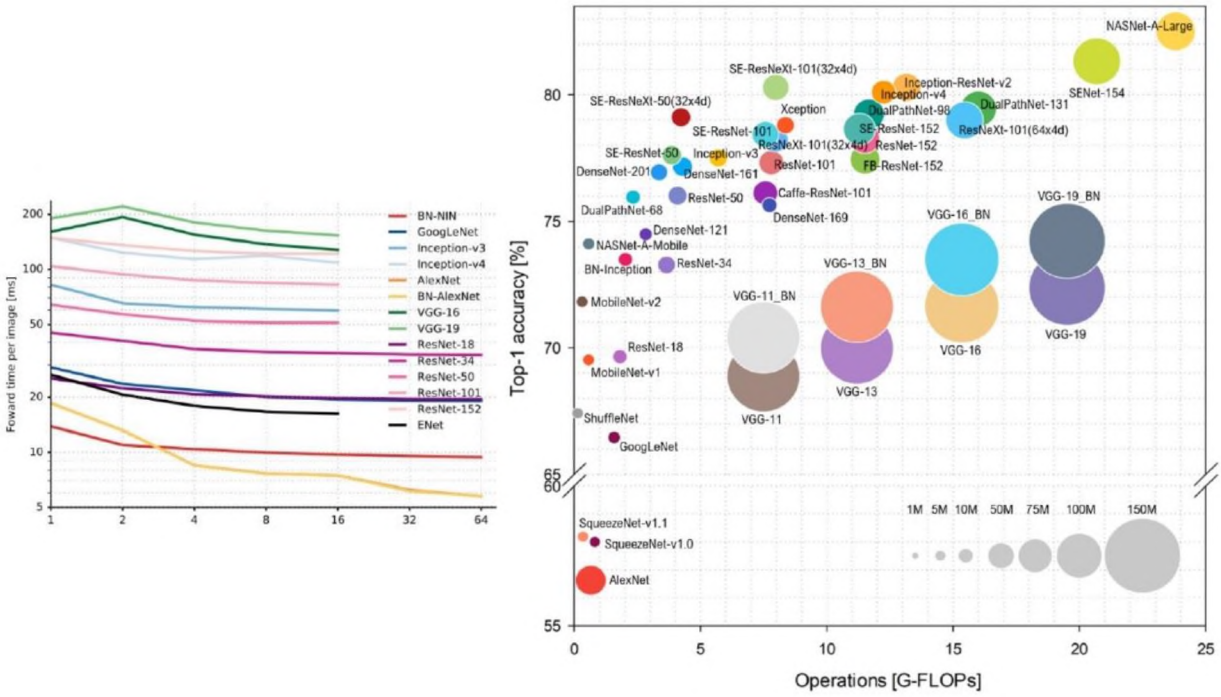

Рисунок 2.2 – Порівняння параметрів CNN (2018 р.)

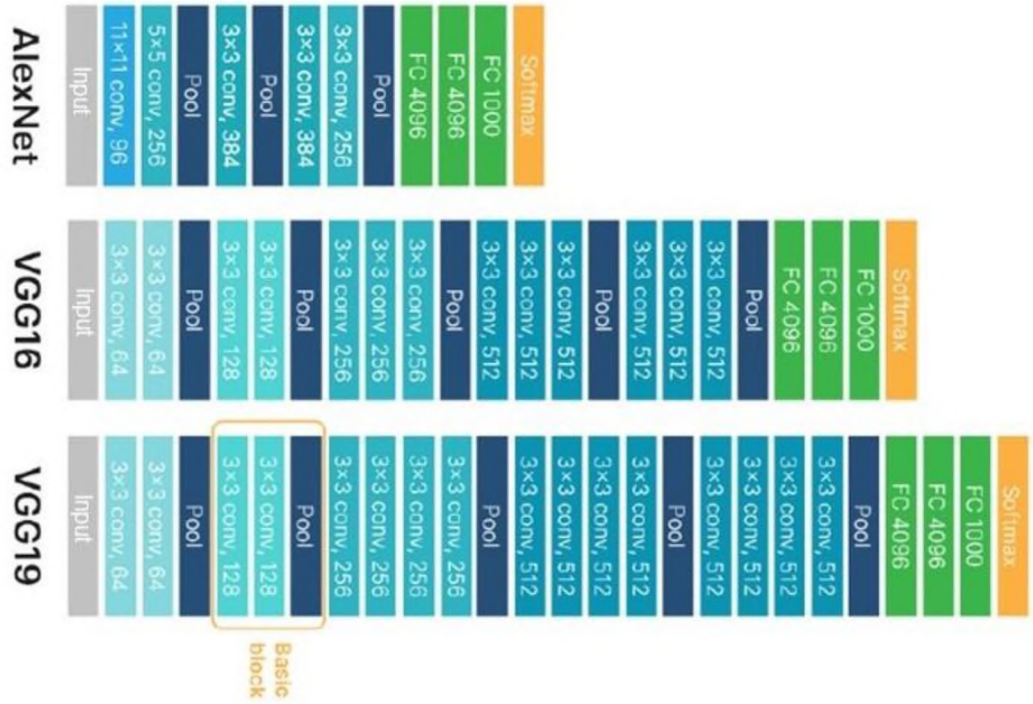

Рисунок 2.3 – Архітектури CNN: AlexNet [33], VGG16, VGG19

VGG можна охарактеризувати наступним чином. Однорідність. Мережа складається з послідовності згорткових шарів, що йдуть один за одним. Ці шари, як правило, мають фільтри згортки розміром 3×33×3 з кроком 1 і доповненням нулями (padding) для збереження просторових

розмірів. Після кожної групи згорткових шарів слід шар пулінгу (зазвичай максимального пулінгу) зменшення просторових розмірів. Глибина. Було запропоновано кілька варіантів архітектури VGG, включаючи VGG-16 [34] (з 16 зваженими шарами) та VGG-19 (з 19 зваженими шарами). Число в назві вказує на загальну кількість шарів з вагами (згорткові повнозв'язні). Повнозв'язні шари [35]. Наприкінці мережі розташовані три повнозв'язні шари. Перші два мають по 4096 нейронів, а останній шар відповідає числу класів і зазвичай має 1000 нейронів (для завдання ImageNet). ReLU активація. Як функцію активації в мережі використовується ReLU. VGG стала популярною через свою простоту та ефективність. Однак вона також відома тим, що вимагає багато пам'яті та обчислювальних ресурсів через велику кількість параметрів, особливо у повнозв'язних шарах. Незважаючи на те, що з тих пір були розроблені більш ефективні архітектури, такі як ResNet [36] і EfficientNet [37], VGG досі використовується в багатьох додатках, особливо як основа для перенесення навчання (Transfer Learning). В цілому, вони вимагають великих обчислювальних ресурсів, тому зараз їх має сенс використовувати хіба що у навчальних цілях, а моделі на базі ResNet (ResNet-50, ResNet-152) досить гарні в сенсі точності якогось великого відриву від них (рис. 2.4). Ця архітектура у 2015 р. виграла змагання ImageNet Large Scale Visual Recognition Challenge (ILSVRC), зробивши справжню революцію глибини нейронних мереж. Вона складалася зі 152 шарів і знизила відсоток помилок до 3,57 %. Це зробило її майже вдвічі ефективніше GoogleNet. ResNet (Residual Network) – це архітектура глибокої нейронної мережі, яка була представлена у 2015 р. на конференції NeurIPS у роботі Kaiming He та його колег із к. Microsoft Research. Основною інновацією ResNet є введення «залишкових блоків», які дозволяють навчати набагато глибші мережі, ніж це було можливо раніше.

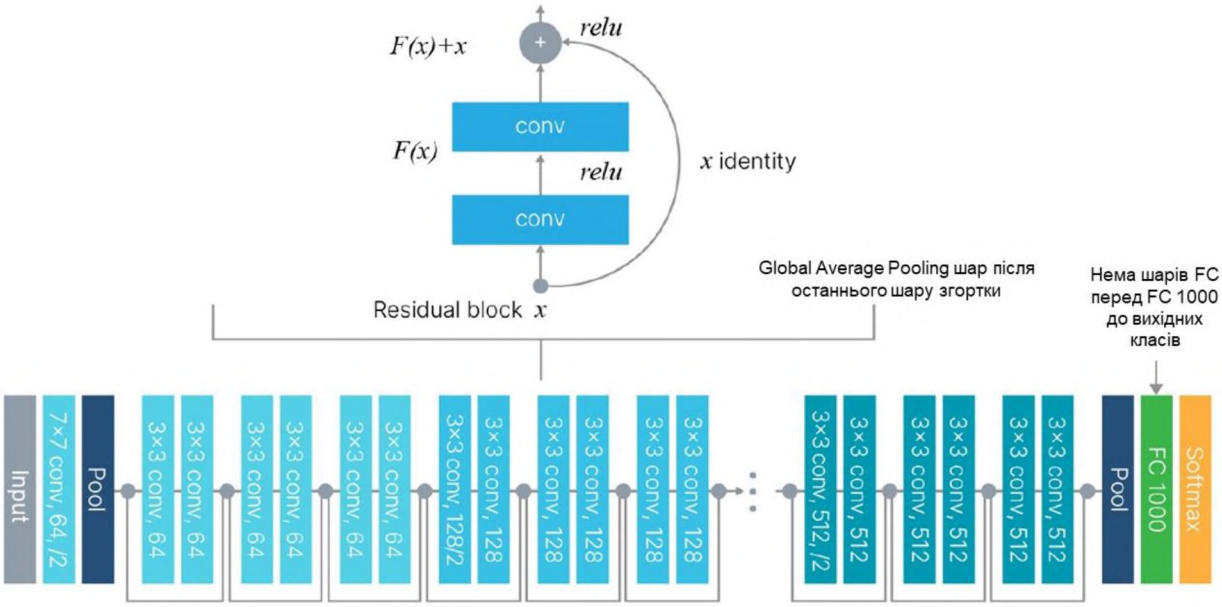

Рисунок 2.4 – Архітектура ResNet-152

Основна ідея ResNet полягає в тому, щоб замість того, щоб на кожному шарі вивчати подання зображення з нуля, вивчати лише різницю (або «залишок») між входом і виходом блоку. Це робиться за допомогою прямих зв'язків (або «наскрізних з'єднань»), які передають інформацію безпосередньо від входу блоку до його виходу. До основних переваг ResNet слід віднести кілька положень.

Протидія загасанню градієнта. У глибоких мережах градієнти можуть згасати або вибухати в процесі зворотного розповсюдження. Наскрізні з'єднання ResNet допомагають градієнтам «протікати» через мережу, роблячи навчання глибоких моделей більш стабільним.

Можливість навчати дуже глибокі мережі. Дослідники успішно навчили мережі ResNet з більш ніж 1000 шарів.

Висока продуктивність. На багатьох завданнях класифікації зображень ResNet показує одні з найкращих результатів.

В результаті цих особливостей ResNet став одним з найбільш популярних типів архітектур, що широко використовуються в області комп'ютерного зору. Але, тим не менш, є моделі, які працюють дещо краще.

DenseNet (2016 г.) – рис. 2.5 [38]. Один із варіантів модифікації існуючих на той час рішень – це додати ще додаткових зв'язків у обхід блоків, щоб градієнт проходив ще краще. Можна замінити суму на конкатенацію.

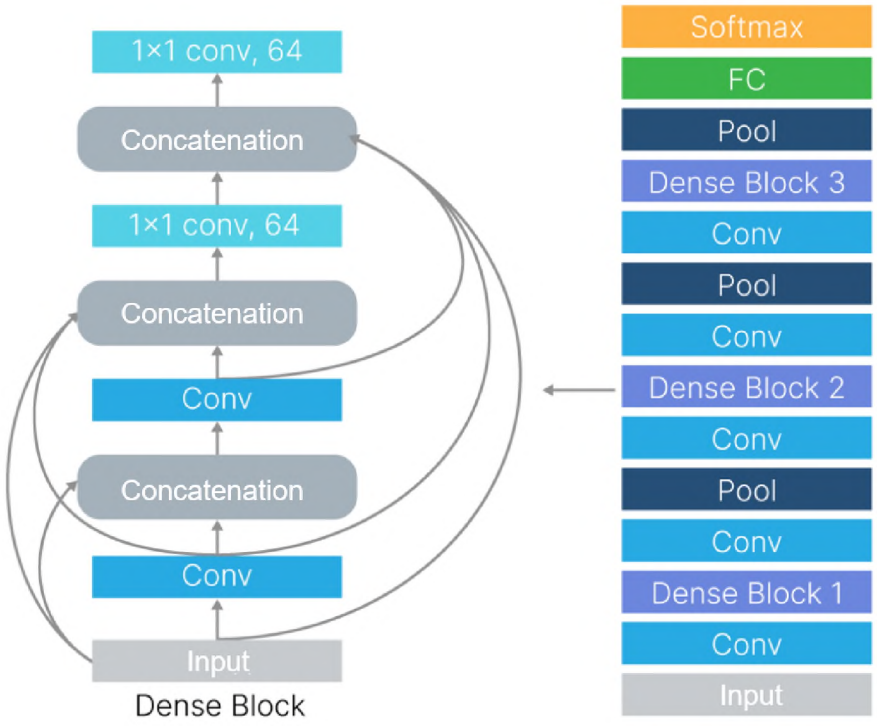

Рисунок 2.5 – Архітектура DenseNet

Це теж працює, але треба розуміти, що конкатенація збільшує кількість ознак. Мабуть, з цим можна боротися за рахунок згорток 1×1. Це так званий DenseNet. З погляду ресурсів він, як правило, трохи більш вимогливий, ніж базовий ResNet і трохи точніше. DenseNet (Densely Connected Convolutional Networks) – це архітектура глибокої згорткової нейронної мережі, представлена у 2016 р. Основна ідея DenseNet полягає в щільній зв'язаності шарів, де кожен шар отримує на вхід усі попередні виходи шарів, утворюючи «щільне» з'єднання. DenseNet мають такі основні характеристики та переваги.

Щільні з'єднання. Ключовою особливістю DenseNet є те, що кожен шар приймає як вхідні дані всі попередні вихідні шари. Ця щільна сполука забезпечує покращене поширення градієнтів під час навчання та зменшує ризик загасання градієнтів.

Ефективне використання параметрів. Щільні з'єднання також забезпечують регуляризацію, що робить модель менш схильною до перенавчання навіть при меншій кількості параметрів порівняно з іншими архітектурами.

Зростання параметрів. Замість збільшення кількості фільтрів після кожного блоку, як це робиться в багатьох інших архітектурах, DenseNet збільшує їх лінійно. Наприклад, якщо перший шар виробляє 4 карти ознак, а «зростання» встановлено на 2, наступний шар вироблятиме 6 карт ознак, наступний 8 і т. д.

Ефективне використання обчислень. Щільні з'єднання забезпечують більшу перевикористання ознак, що робить DenseNet більш ефективним з точки зору обчислень.

Перехідні блоки. Щоб контролювати розмірність та зменшувати просторові розміри, DenseNet використовує так звані «перехідні блоки» між щільно зв'язаними блоками. Ці блоки зазвичай містять згортковий шар і шар average pooling. DenseNet була однією з перших архітектур, яка показала, що глибокі мережі можуть бути покращені не лише поглибленням, а й посиленням внутрішніх з'єднань між шарами.

Незважаючи на свої переваги, DenseNet може бути обчислювально насиченою через щільні з'єднання, особливо в дуже глибоких мережах.

WideResNet (Wide Residual Networks) появилась у 2016 р. Це модифікація ResNet, де збільшується ширина кожного шару, щоб покращити продуктивність моделі без збільшення її глибини. Основна ідея WideResNet вже не з розпаралелюванням, а полягає в тому, щоб посилити мережу, збільшуючи кількість каналів (або фільтрів) у згорткових шарах (рис. 2.6). Тобто, в ній можемо збільшувати кількість фільтрів і зменшити кількість шарів. Також варто відзначити що 50 шаровий Wide ResNet показує кращі результати ніж 52-их шаровий ResNet. З погляду обчислювальних ресурсів використання ширини замість глибини – ефективніше (parallelizable). Параметр «Widening Factor» (k) – параметр визначає, у скільки разів

збільшується кількість фільтрів у кожному шарі порівняно з базовою ResNet. Наприклад, якщо базової ResNet 16 фільтрів і $k = 2$, то WideResNet буде 32 фільтра.

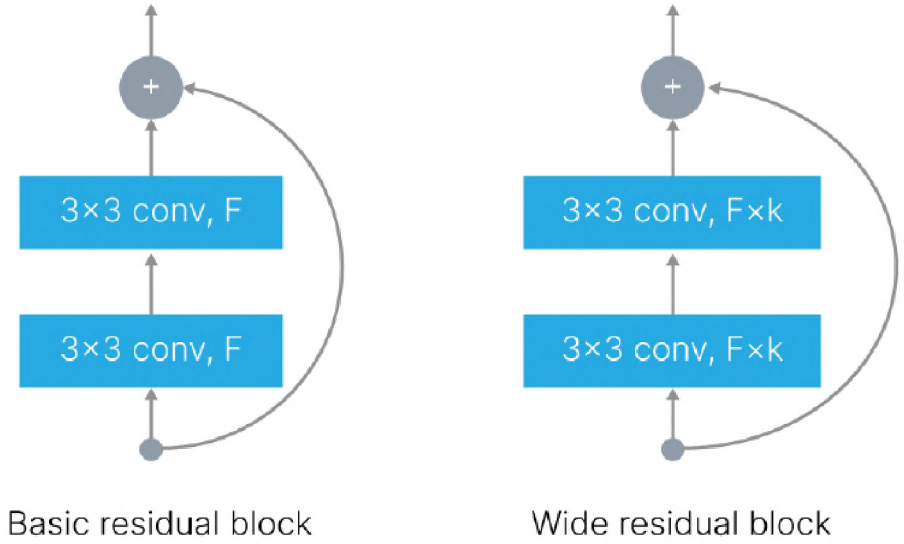

Рисунок 2.6 – Порівняння базового ResBlock та широкого ResBlock, де використовуються F×k фільтри замість F фільтрів у кожному шарі

Збільшення ширини зазвичай покращує продуктивність мережі при помірному збільшенні числа параметрів та обчислювальної складності. у таких моделях перша цифра – кількість шарів, друга – коефіцієнт, з яким збільшуємо кількість фільтрів. Автори стверджують, що самі по собі Residuals значно важливіший фактор, ніж глибина. Таким чином, до переваг WideResNet слід віднести покращену продуктивність при порівнянні або навіть меншій кількості параметрів у порівнянні з глибокими ResNets і скорочення часу на навчання, оскільки має меншу глибину. Недоліками WideResNet є збільшення ширини, що може призвести до збільшення кількості параметрів, яке може потребувати більше пам'яті та обчислювальної потужності. В цілому, WideResNet пропонує цікавий компроміс між глибиною і шириною мережі, дозволяючи досягати високої продуктивності за порівнянної обчислювальної складності.

2.3 Вибір метрики для оцінки продуктивності моделі глибокого навчання при дисбалансі класів

Проблема незбалансованих класів полягає в тому, що традиційні метрики, такі як точність (Accuracy), можуть вводити в оману, демонструючи високу продуктивність моделі, навіть якщо вона просто передбачає клас, що найчастіше зустрічається [39]. Як наслідок, вибір правильної метрики оцінки для незбалансованих датасетів стає важливим для створення ефективних і надійних моделей [40].

Recall (повнота) – це метрика, яка вимірює, наскільки добре модель може ідентифікувати всі позитивні випадки в даних. Вона обчислюється як відношення істинно позитивних прогнозів до суми істинно позитивних і хибно негативних прогнозів:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}. \quad (2.1)$$

У контексті незбалансованих класів, де один клас переважає над іншим (наприклад, 99 % проти 1 %), простий вимір *Recall* може не давати повного уявлення про продуктивність моделі. У таких випадках може знадобитися використовувати балансовані метрики або враховувати продуктивність окремо для кожного класу. При цьому, доцільно використовувати кілька метрик та поглядів на дані, щоб отримати повне уявлення про продуктивність вашої моделі.

Міра *F*1. У деяких завданнях однаково важливо і добре ідентифікувати позитивні приклади (висока повнота) і мінімізувати кількість помилкових спрацьовувань (висока точність). *F*1-міра допомагає поєднати ці дві метрики в одну, що особливо корисно, коли потрібно врівноважити їх. Наприклад, при детекції спаму треба переконатися, що блокуєте якнайбільше спам-листів (висока повнота), але при цьому не хочете помилково блокувати легітимні листи (висока точність). *F*1-міра дозволяє порівнювати різні моделі з урахуванням обох цих завдань. Однак, вона має певні обмеження.

1. $F1$-мера не враховує істинно негативні прогнози (True Negatives), отже вона може підходити для завдань, де негативні класи також важливі.

2. Як і будь-яка інша метрика, $F1$-міра не повинна розглядатися ізольовано. У деяких контекстах може бути корисно розглядати точність, повноту та інші метрики для отримання повного уявлення про продуктивність моделі.

Як відомо, точність (Precision) – це частка істинно позитивних прогнозів серед усіх позитивних прогнозів:

$$Precision = \frac{True\,Positive}{True\,Positive + False\,Positive}. \quad (2.2)$$

В загальному випадку, $F1$-міра має вигляд:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \quad (2.3)$$

Макро-усереднення. Для багатокласової класифікації можна обчислити Precision, Recall або F1-міру для кожного класу окремо, а потім взяти їх середнє значення. Це дозволяє враховувати продуктивність кожного класу однаково, попри їх дисбаланс.

Криві ROC та AUC. Площа під ROC-кривою (AUC) є метрикою, яка може бути особливо корисною для оцінки продуктивності моделі на незбалансованих даних. AUC вимірює здатність моделі розрізняти класи незалежно від граничного значення.

Вагові метрики. Можна привласнити ваги різним класам залежно від їхньої представленості даних. Таким чином, менші класи можуть мати більшу вагу в оцінці продуктивності, щоб компенсувати їхнє менше представлення. Розглянемо деякі приклади. Зважена точність та зважена повнота.

Ці метрики обчислюються шляхом зважування точності та повноти кожного класу на основі їх представленості в датасеті:

$$Weighted\,Precision = \sum_i (weight_i \cdot Precision), \quad (2.4)$$

де $weight_i$ – це частка прикладів класу i.

Зважена $F1$-міра. Аналогічно до (2.4), зважена $F1$-міра враховує важливість кожного класу при обчисленні середнього значення $F1$-міри по всіх класах.

Вартість помилок. У деяких додатках певні помилки можуть мати вищу вартість порівняно з іншими. Наприклад, при медичній діагностиці помилковонегативне передбачення може мати набагато серйозніші наслідки, ніж помилковопозитивне передбачення. У таких ситуаціях можна використовувати матрицю вартості помилок для зважування різних типів помилок.

Sample weights. У задачах регресії або класифікації іноді одним прикладам надається більша вага порівняно з іншими. Це може бути корисним, якщо деякі приклади вважаються більш надійними, репрезентативними або важливими з будь-яких причин. Більшість алгоритмів машинного навчання дозволяють задати ваги окремих прикладів у процесі навчання.

Зважування функції втрат. У деяких завданнях, особливо при глибокому навчанні, можна модифікувати функцію втрат, вводячи ваги для різних класів або прикладів. Це може допомогти моделі краще зосередитися на рідко зустрічаються або важливіших класах.

Balanced Recall є альтернативою звичайної чутливості, особливо коли класи в датасеті незбалансовані. Ця метрика обчислюється як середня арифметична *Recall* (або точність) для кожного класу. Наприклад, формула для бінарної класифікації має вигляд:

$$Balanced\ Recall = \frac{1}{2}(Recall_{позитивний\ клас} + Recall_{негативний\ клас}). \qquad (2.5)$$

У разі бінарної класифікації балансована точність є середнім значенням чутливості (*Recall* для позитивного класу) і специфічності (*Recall* для негативного класу). Наприклад, якщо є матриця помилок, що відповідає рис. 2.7. *Recall* для позитивного класу відповідає виразу:

$$Recall_{\text{позитивний клас}} = \frac{50}{50+10} = \frac{50}{60} = 0{,}833. \quad (2.6)$$

Специфіка (*Recall* для негативного класу) має вигляд:

$$Recall_{\text{негативний клас}} = \frac{35}{35+5} = \frac{35}{40} = 0{,}875. \quad (2.7)$$

Тепер, використовуючи формулу для *Balanced Recall*, отримаємо:

$$Balanced\ Recall = \frac{1}{2}(0{,}833 + 0{,}8755) = 0{,}8542. \quad (2.8)$$

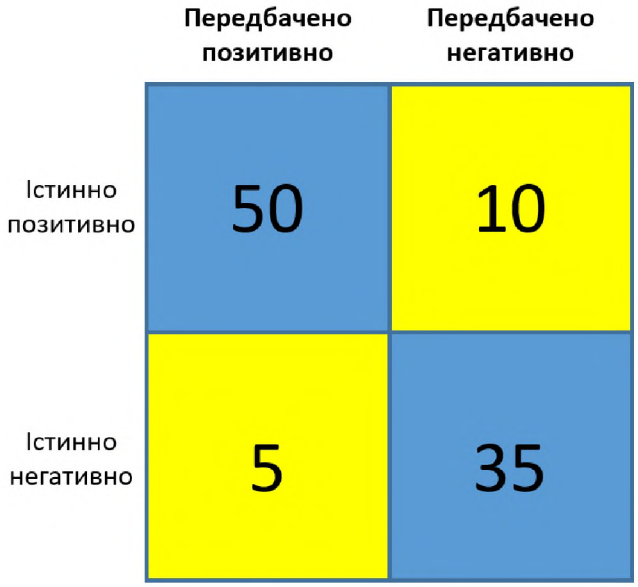

Рисунок 2.7 – Матриця помилок

На даний час, існує кілька випадків, що потребують використання балансованої точності: незбалансовані дані та більш повне представлення про те, як модель працює на кількох класах, а не лише на одному з них.

Таким чином, використання вагових метрик дозволяє враховувати особливості датасету та специфіку проблеми, яку намагаєтесь вирішити. Однак важливо пам'ятати, що додавання ваги може ускладнити інтерпретацію результатів, тому завжди варто уважно перевіряти, як введення ваги впливає на продуктивність моделі глибокого навчання.

2.4 Реалізація Transfer Learning

Як відомо, передача навчання (Transfer Learning) у контексті CNN – це метод навчання моделі на одній задачі та використання отриманих знань та параметрів для вирішення іншої, пов'язаної задачі. Основна перевага цього методу полягає у можливості використовувати переднавчені моделі, які вже навчені на великих обсягах даних та вивчили високорівневі ознаки, які можуть бути узагальнені та застосовані до нового завдання. На даний час, до основних переваг Transfer Learning CNN слід віднести наступне.

Ефективне використання даних. Передбачені моделі на великих наборах даних, таких як ImageNet, вже вивчили загальні характеристики, такі як грані, текстури, форми тощо, що дозволяє використовувати їх для завдань, де доступна обмежена кількість даних.

Скорочення часу навчання. Найчастіше навчання CNN з нуля великих наборах даних то, можливо ресурсо- і часо-витратним процесом. Передача навчання дозволяє значно скоротити час навчання, оскільки модель вже має деякі знання.

Поліпшення узагальнення. Передбачені моделі, особливо якщо вони навчені на різноманітних даних, мають здатність краще узагальнювати зображення та ознаки, що може призвести до більш стійких та точних моделей.

Менша потреба даних. Передача навчання дозволяє створити високоякісні моделі, навіть якщо у вас немає великої кількості розмічених даних для конкретного завдання.

Розв'язання специфічних завдань. Шляхом донавчання останніх шарів CNN можна адаптувати модель до конкретної задачі, що особливо корисно у разі коли основна архітектура моделі підходить, але потрібно навчити модель на конкретних даних.

В ході досліджень визначено основні кроки, які зазвичай виконуються при застосуванні Transfer Learning до CNN.

1. Вибір попередньо вивченої моделі. Виберіть попередньо вивчену модель CNN, яка була навчена на великому наборі даних для схожого завдання. Залежно від завдання та даних, які маєте, Ви можете вибрати, наприклад, модель ResNet, VGG, Inception [41], MobileNet [42] або іншу відповідну модель.

2. Заморожування вагів. Після вибору моделі заморозьте (заблокуйте) ваги всіх або частини шарів моделі. Це означає, що ваги цих верств не оновлюватимуться в процесі навчання на новому завданні. Однак це необов'язковий крок, і в залежності від ваших даних та задачі Ви можете вирішити залишити деякі шари розмороженими і навіть донавчити їх.

3. Заміна вихідного шару. Замініть вихідний шар попередньо вивченої моделі таким чином, щоб він відповідав числу класів або цільовому вихідному формату нового завдання. Цей новий вихідний шар буде випадково ініціалізований і його ваги підлягатимуть навчанню.

4. Навчання моделі. Навчіть модель на своїх нових даних, використовуючи попередньо навчені ваги в заморожених шарах та навчаючи лише вихідний шар. Ви можете використовувати метод зворотного розповсюдження помилки (backpropagation) та оптимізатор для мінімізації функції втрат на вашому новому завданні.

5. Навчання з розмороженими шарами (опціонально). Якщо у вас є достатньо даних, Ви можете вирішити розморозити кілька верхніх шарів попередньо навченої моделі та донавчити їх на ваших даних. Це може покращити продуктивність моделі.

6. Оцінка та налаштування: Оцініть продуктивність моделі на валідаційному наборі даних. Можливо, вам знадобиться налаштувати параметри моделі, такі як швидкість навчання (learning rate) та архітектуру, щоб досягти кращих результатів.

7. Тестування: Після налаштування моделі протестуйте її на тестовому наборі даних, щоб оцінити її продуктивність у реальних умовах.

Таким чином, Transfer Learning використовується для підвищення ефективності на невеликих датасетах за рахунок перенесення ваг з великих датасетів, а також дозволяє значно прискорити навчання моделі та досягти хороших результатів, особливо за наявності обмеженої кількості даних для нового завдання. З іншого боку, підвищуючи ефективність, Transfer Learning іноді вносить частку небажаної необ'єктивності.

На основі проведених досліджень можна сформувати програмний код на мові Python, наприклад, для нейронної мережі попередньо навченої мережі VGG16, яку необхідно додати до певної базової архітектури (рис. 2.7).

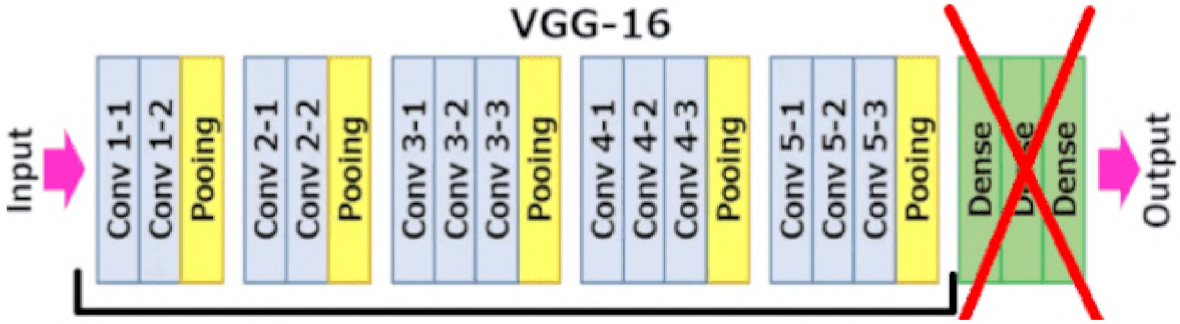

Рисунок 2.7 – Принцип використання попередньо навченої VGG16

Для початку завантажуємо необхідні бібліотеки [43]:

```
import tensorflow as tf                          # Бібліотека TensorFlow
from keras.utils.vis_utils import plot_model   # метод для малювання моделі
import gc   # Модуль Garbage Collector - для допомоги у звільненні пам'яті
              # (Видаляє зайві/не використовувані дані)
import os   # Модуль для роботи з файлами операційної системи
import time                          # Модуль для роботи з часом
import gdown                         # Модуль для завантаження великих файлів
import random                        # Модуль для створення випадкових значень
import seaborn as sns                # Модуль для роботи з графіками та стилями
import matplotlib.pyplot as plt # Модуль для роботи з графіками, зображеннями
from PIL import Image                # Методи роботи із зображеннями
from tensorflow.keras import utils # Будемо використовувати цю бібліотеку
from google.colab import files # Імпортуємо бібліотеку files
import pandas as pd             # Бібліотека для роботи з датафреймом
import numpy as np
from skimage.transform import resize # робота із зображеннями
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Дуже важливо заморозити основу згортки (ваги попередньо навченої мережі), інакше, вони будуть змінюватися. Згорткова основа CNN моделей має властивість інваріатності, тобто, не запам'ятовує розташування об'єктів, на відміну повнозв'язних шарів:

```
# Створюємо базову модель VGG16:
# Не вмикаємо верхню частину моделі, щоб можна було використовувати свої
# вхідні розмірності (запис include_top=False) і використовуємо ваги вже
# попередньої моделі на датасеті Imagenet
base_model = tf.keras.applications.vgg16.VGG16(input_shape=image_shape,
                                               include_top=False,
                                               weights='imagenet')
```

Таким чином, надалі збираємо повну модель, що складається з усіх певних шарів базової архітектури

```
preprocess_input = tf.keras.applications.vgg16.preprocess_input
# Нормалізація даних на вхід відповідно до вимог моделі
gc.collect()
# Складання сміття для звільнення пам'яті
image_shape = image_size + (3,)
# Визначаємо вхідну розмірність відповідно до вимог моделі
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
# Задаємо шар GlobalAveragePooling2D
prediction_layer = tf.keras.layers.Dense(len(class_names))
# Задаємо вихідний шар на потрібну нам кількість класів
```

Створюємо базову модель VGG16:

```
base_model = tf.keras.applications.vgg16.VGG16(input_shape=image_shape,
                            include_top=False,
                            weights='imagenet')
# Не вмикаємо верхню частину моделі, щоб можна було використовувати
# свої вхідні розмірності
# Використовуємо ваги попередньої моделі на базі Imagenet
```

Збираємо повну модель, що складається з усіх наших певних шарів та базової моделі VGG16:

```
inputs = tf.keras.Input(shape=(image_shape))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Flatten()(x)
# Додаємо вирівнюючий шар, кілька повнозв'язних шарів на 256 нейронів,
# шар нормалізації та Dropout
x = tf.keras.layers.Dense(256)(x)
x = tf.keras.layers.BatchNormalization()(x)
```

```
x = tf.keras.layers.Dense(256)(x)
x = tf.keras.layers.Dense(256)(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)

base_learning_rate = 0.0001 # Задаємо значення кроку навчання
epochs = 100                # Задаємо кількість епох

# Компілюємо модель:
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_
rate),
# Оптимізатор Adam із заданим раніше кроком навчання

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
# Помилка SparseCategoricalCrossentropy та метрика Accuracy

callback_MC = ModelCheckpoint('C:/Users/sulim/Documents/ResNet50_fin.h5',
                              monitor='val_loss',
                              verbose=1,
                               save_best_only=True)
# Записуємо колбек для збереження епохи з найкращою точністю

earlystop = EarlyStopping(monitor='val_loss', verbose=1, patience=5)
# Записуємо колбек для зупинки навчання, якщо помилка на перевірочній вибірці
# стала погіршуватися
```

```
CurrentTime = time.time()
# Засікаємо час для вимірювання часу навчання нейронної мережі
gc.collect()
```

Надалі виконуємо донавчання моделі:

```
VGG16 = model.fit(train_ds,
                  epochs=epochs,
                   validation_data=val_ds,
                     callbacks=[callback_MC, reduce_lr, earlystop])
# Навчальна вибірка і кількість епох навчання (задана раніше),
# перевірочна вибірка і вказуємо колбеки
plot_graph(VGG16.history['accuracy'],        # Виводимо графіки навчання
           VGG16.history['val_accuracy'],
           VGG16.history['loss'],
           VGG16.history['val_loss'])

print('Час навчання: {:.0f} секунд ({:.2f} хвилин)'.format(time.time() -
CurrentTime, (time.time() - CurrentTime)/60))  # Показуємо час навчання
```

Аналогічним чином, можна використовувати інші попередньо навчені нейронні мережі.

Висновки до розділу 2

Щоб дослідити вплив дисбалансу датасету на продуктивність нейронної мережі, запропоновано сформувати кілька датасетів на базі CIFAR-10, який містить 10 класів. Новостворені набори відрізняються рівнем дисбалансу між класами. Сам дисбаланс описується фіксованою послідовністю та має кілька параметрів. Розмір навчальної та перевірочної вибірок і максимально досяжний рівень дисбалансу вибирається з урахуванням збереження адекватності синтезованої моделі глибокого навчання нейронної мережі класифікації зображень.

З метою реалізації класифікації зображень здійснений вибір архітектури нейронної мережі. В якості базової розглядається згорткова нейронна мережа. В ході досліджень проаналізовані особливості CNN з архітектурами VGG, ResNet і DenseNet та ін.

Враховуючі наявність незбалансованих класів, традиційні метрики для оцінки продуктивності мережі використовувати не доцільно через їх вплив на ефективність і надійність моделей глибокого навчання класифікації зображень. Крім того, у деяких завданнях однаково важливо і добре ідентифікувати позитивні приклади (висока повнота) і мінімізувати кількість помилкових спрацьовувань (висока точність). Як наслідок, в роботі досліджено застосування балансованих метрик або враховування продуктивності окремо для кожного класу.

Основна увага приділяється *Balanced Recall* є альтернативою звичайної чутливості, особливо коли класи в датасеті незбалансовані. Ця метрика обчислюється як середня арифметична *Recall* (або точність) для кожного класу.

В свою чергу, при використанні вагових метрик важливо брати до уваги, що додавання ваги може ускладнити інтерпретацію результатів, тому завжди варто уважно перевіряти, як введення ваги впливає на продуктивність моделі.

З метою вивчення впливу дисбалансу датасету в роботі запропоновано використовувати Transfer Learning. Такий підхід дозволяє ефективно використовувати дані; скоротити час навчання; поліпшити узагальнення; знизити вимоги до обсягу початкової бази даних; вирішувати специфічні завдання. Для реалізації Transfer Learning досліджено основні етапи застосування попередньо навчених CNN у синтезованій моделі глибокого навчання нейронної мережі класифікації зображень.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

3.1 Оцінка впливу дисбалансу на точність класифікації нейронної мережі

Згідно п. 2.2, проведено дослідження впливу дисбалансу датасету на продуктивність моделей (рис. 3.1), що містять попередньо навчені мережі VGG16, ResNet152v2, DensNet169. На першому етапі використаний датасет C10img4000full, в якому кількість зображень для кожного з 10-ти класів однакова, тобто, відсутній дисбаланс.

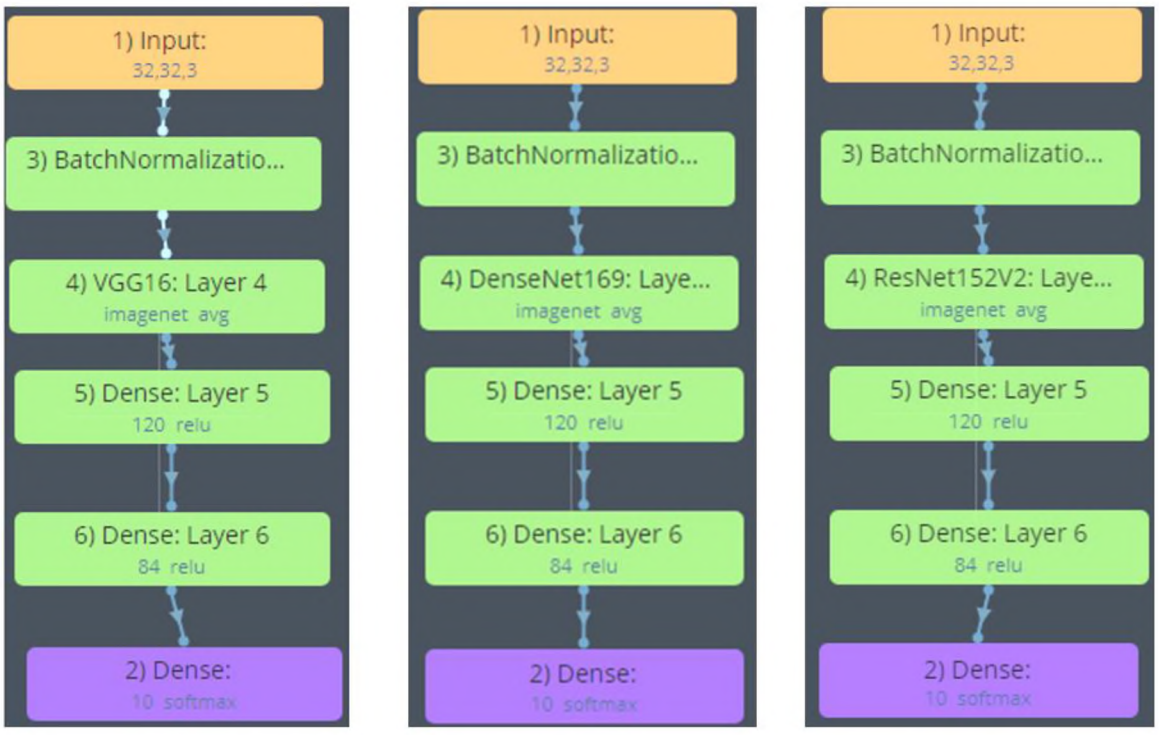

Рисунок 3.1 – Модель глибокого навчання на основі VGG16, ResNet152v2, DensNet169

Отримані оцінки представлені у вигляді графіків, де по Ох – кількість епох навчання, а по Оу – відповідна метрика. При цьому, для кожного

варіанту кількість епох навчання визначалась окремо, щоб уникнути ситуації перенавчання моделі.

Для моделі з VGG16 метрика *Balanced Recall* представлена на рис. 3.2, а матриця помилок – рис. 3.3. На навчальній вибірці модель ідентифікувала всі позитивні випадки в даних на рівні 100 %, на валідаційній – 86%.

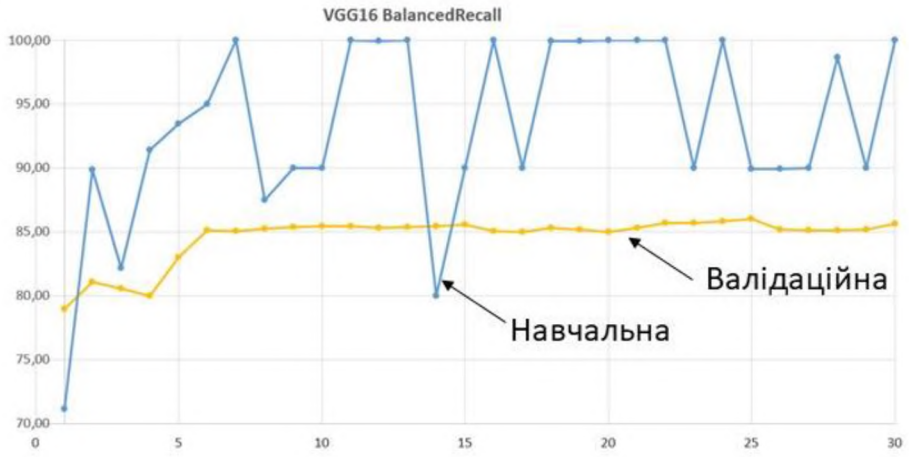

Рисунок 3.2 – *Balanced Recall* для моделі з VGG16

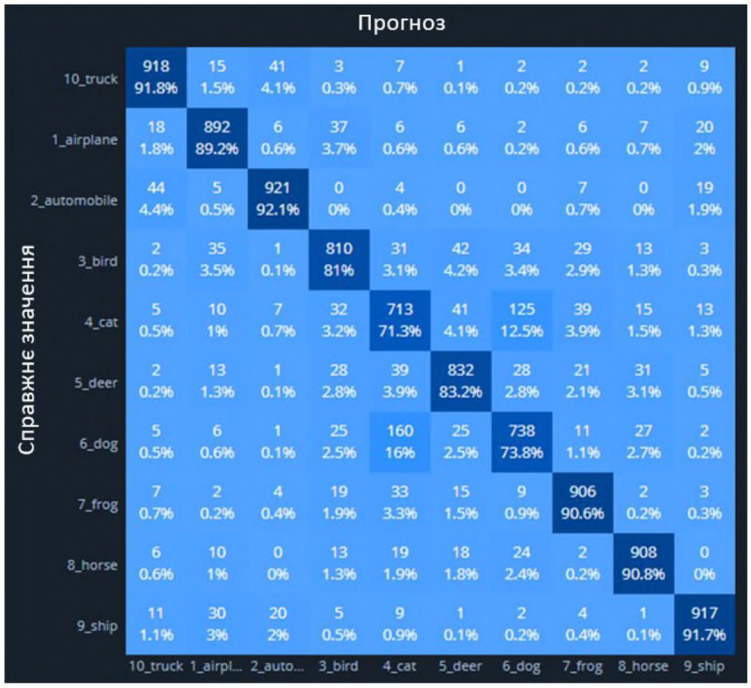

Рисунок 3.3 – Матриця помилок для моделі з VGG16

Аналогічно отримано результати моделювання для варіанту ResNet152v2 (рис. 3.4). На навчальній вибірці модель ідентифікувала всі позитивні випадки в даних на рівні 97,5 %, на валідаційній – 75,1 %.

Для моделі з DensNet169 на навчальній вибірці ідентифікувала всі позитивні випадки в даних на рівні 100 %, на валідаційній – 84,2 % (рис. 3.5).

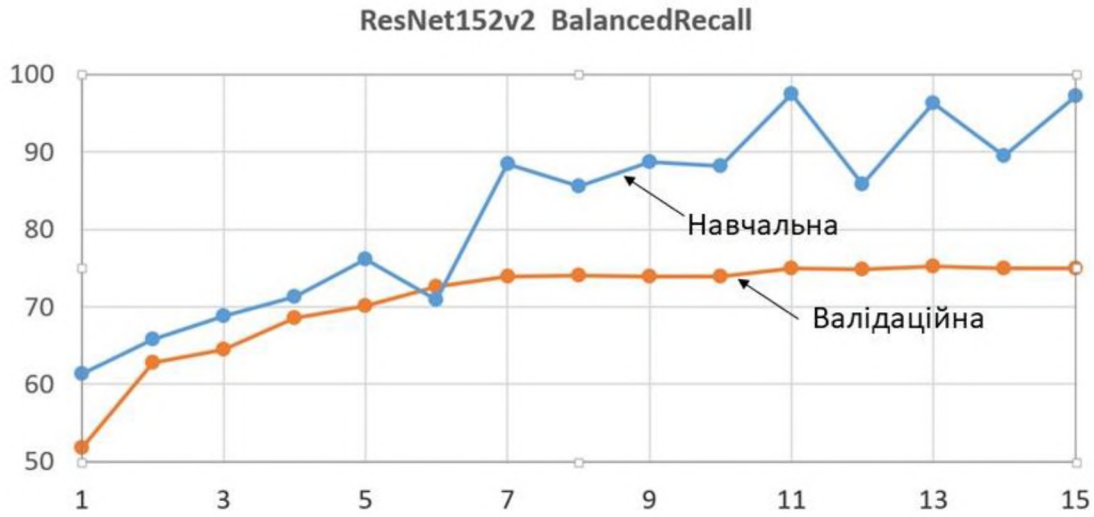

Рисунок 3.4 – *Balanced Recall* для моделі з ResNet152v2

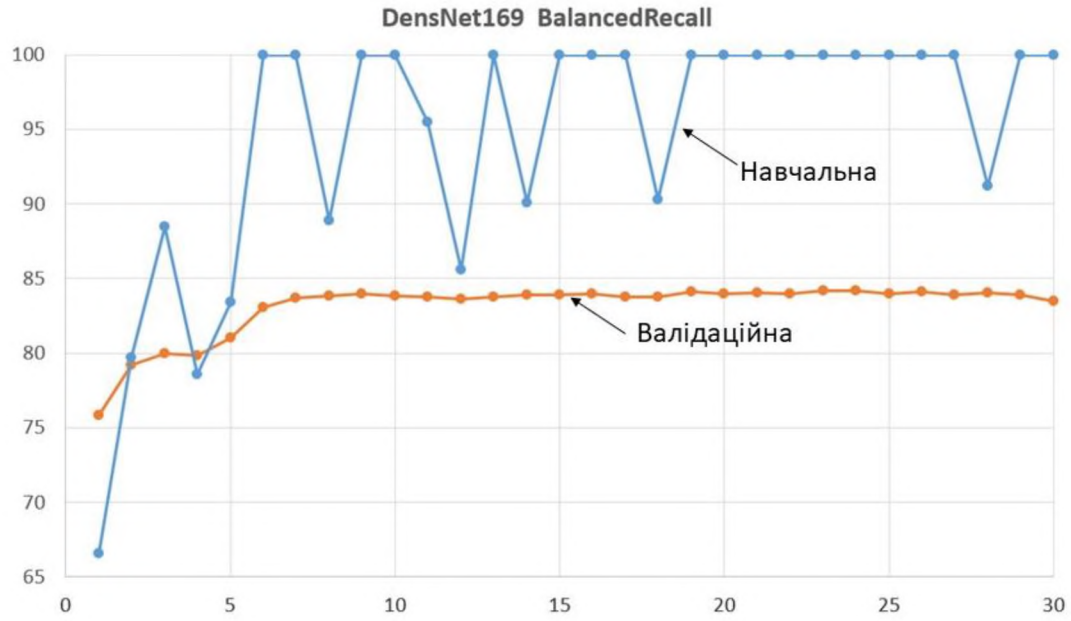

Рисунок 3.5 – *Balanced Recall* для моделі з DensNet169

Отримані результати свідчать, що хоча модель з VGG16 має найбільшу продуктивність за метрикою *Balanced Recall*, варіант з DensNet169 виглядає більш привабливим з точки зору вимог до обчислювальних ресурсів і обсягу пам'яті. Однак щоб зробити остаточні висновки на користь того чи іншого підходу треба проаналізувати *Balanced Recall* окремому по кожному з класів.

Так, для моделі з VGG16 найгірше ідентифікується клас № 4 (рис. 3.6). Це свідчить про необхідність удосконалення датасету C10img4000full. Для ResNet152v2 та DensNet169 спостерігається розшарування за певними класами (рис. 3.7 і 3.8).

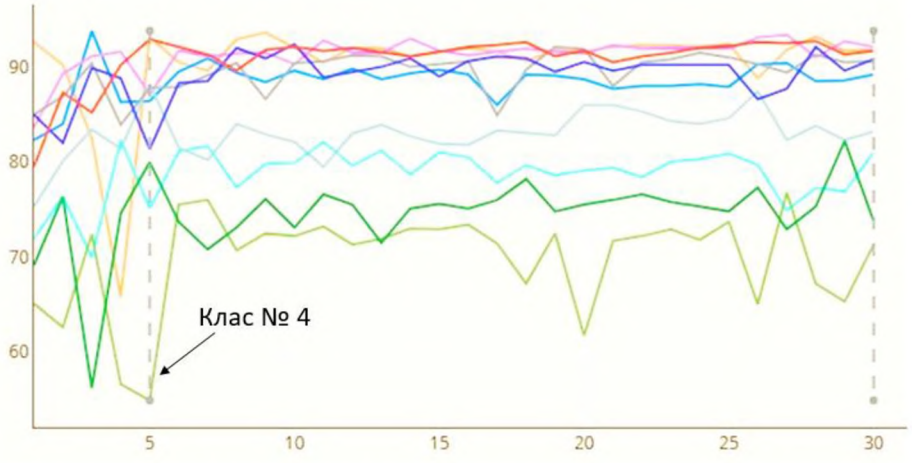

Рисунок 3.6 – *Balanced Recall* по класах для моделі з VGG16

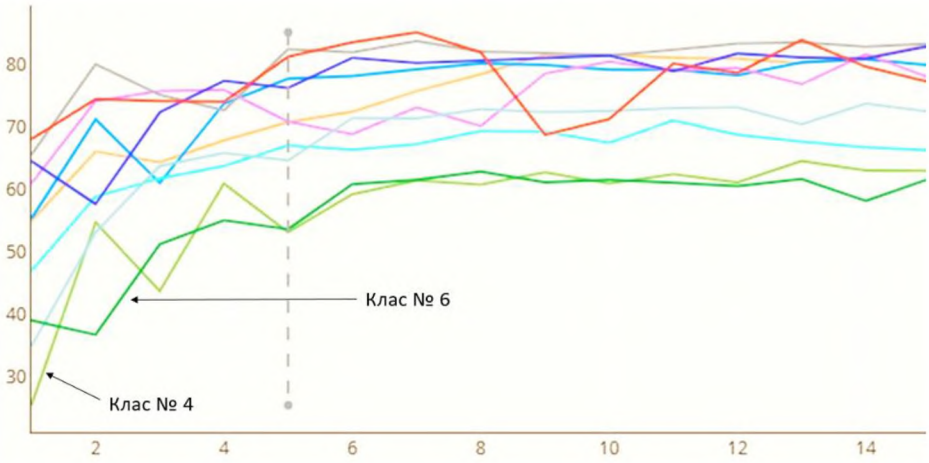

Рисунок 3.7 – *Balanced Recall* по класах для моделі з ResNet152v2

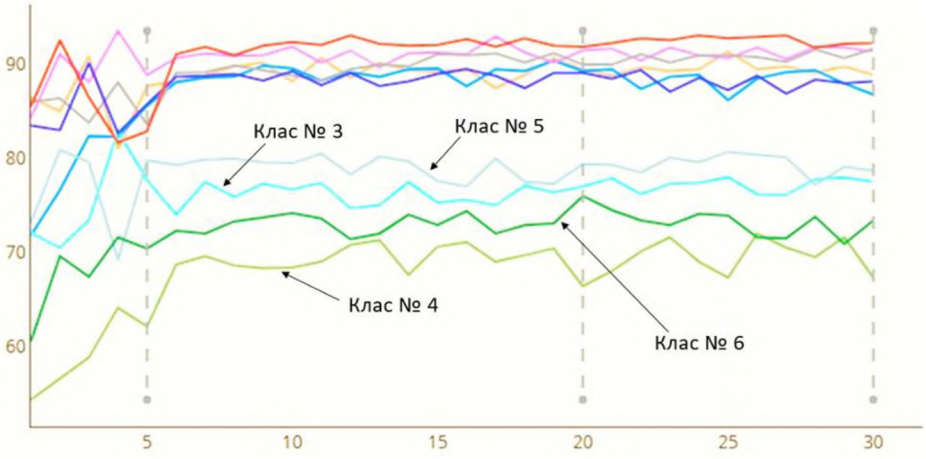

Рисунок 3.8 – *Balanced Recall* по класах для моделі з DensNet169

На другому етапі використаний датасет C10img65 з дисбалансом, що характеризується параметрами $D1$ = 65,725 % і $D2$ = 77,112 % (див. рис. 2.4). На навчальній вибірці модель з VGG16 ідентифікувала всі позитивні випадки в даних на рівні 100 %, на валідаційній – 84 % (рис. 3.9). При цьому кількість епох навчання збільшилась до 50.

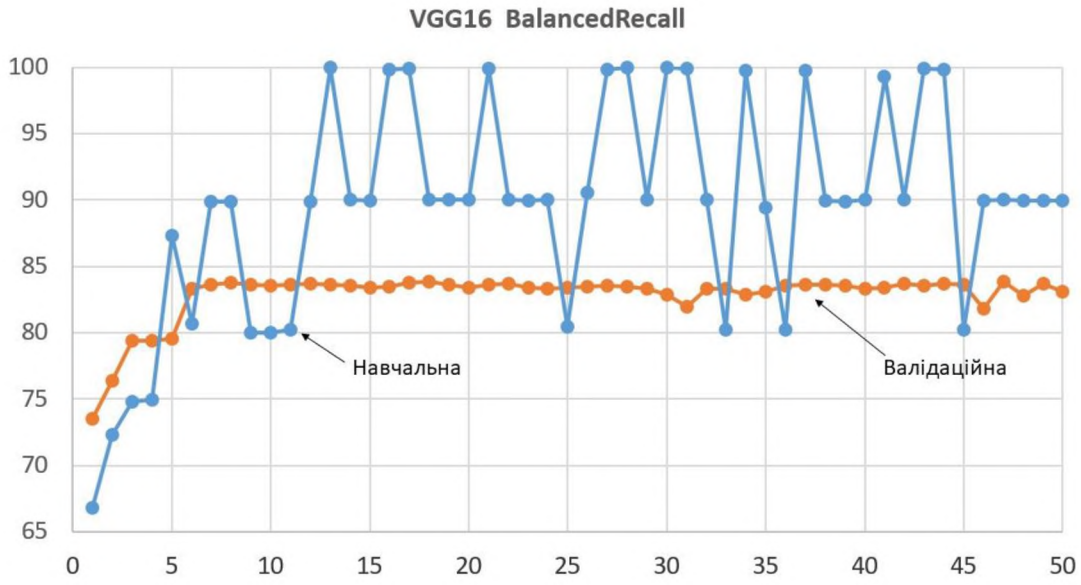

Рисунок 3.9 – *Balanced Recall* для моделі з VGG16 (C10img65)

Відповідно, для моделі з ResNet152v2 ці характеристики мають значення: 100 % і 75 % (рис. 3.10), з DensNet169: 100 % і 84% (рис. 3.11).

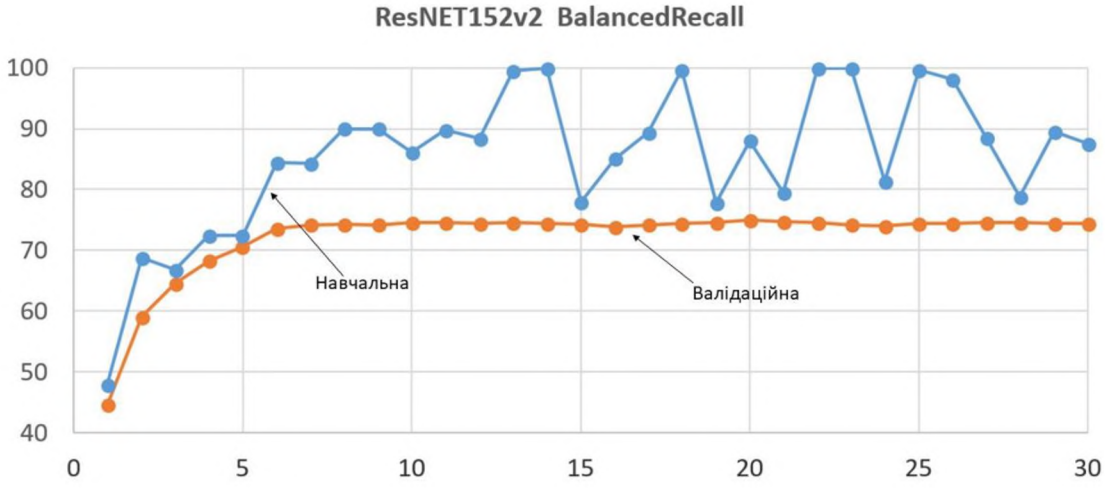

Рисунок 3.10 – *Balanced Recall* для моделі з ResNet152v2 (C10img65)

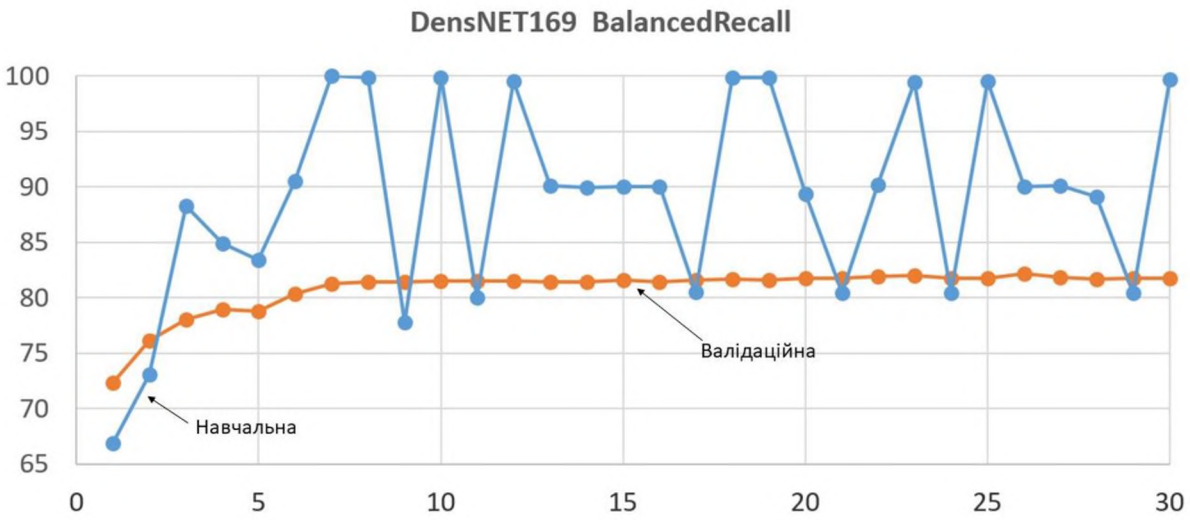

Рисунок 3.11 – *Balanced Recall* для моделі з DensNet169 (C10img65)

Аналогічно, отримано оцінки по окремих класах:

– VGG16 (C10img65) – рис. 3.12;
– ResNet152v2 (C10img65) – рис. 3.13;
– DensNet169 (C10img65) – рис. 3.14.

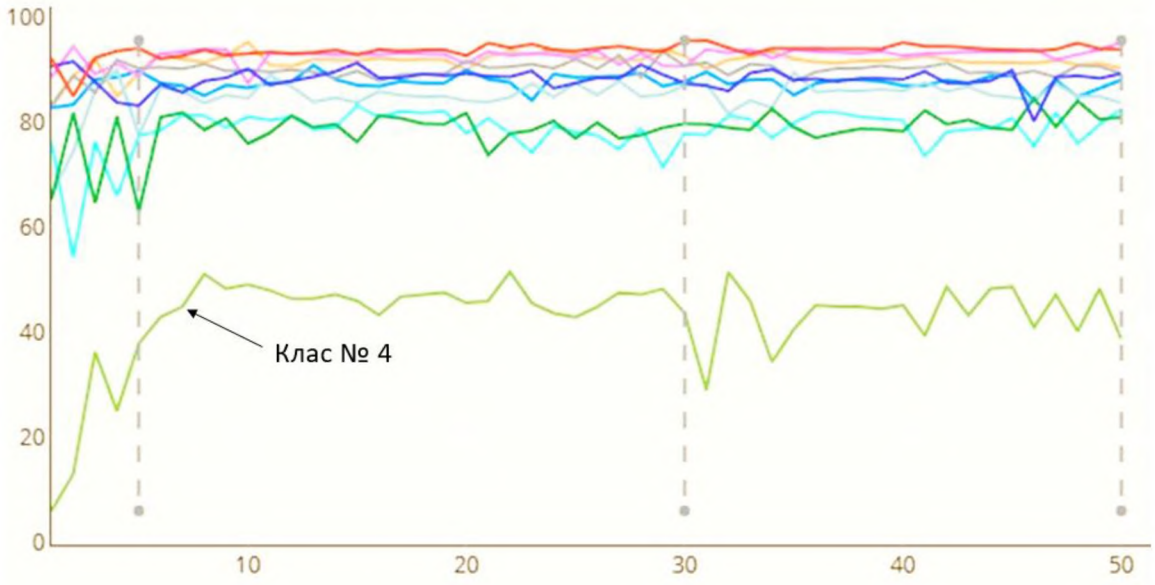

Рисунок 3.12 – *Balanced Recall* по класах для VGG16 (C10img65)

Як і очікувалось, найгірша продуктивність моделі відповідає класу № 4, який має найменшу кількість зображень у датасеті.

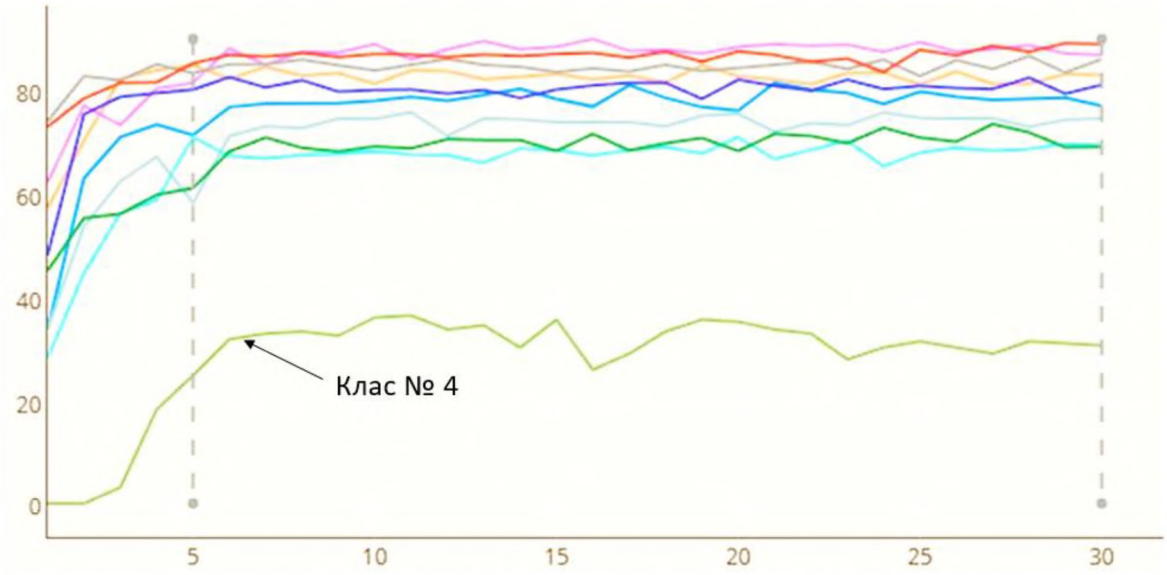

Рисунок 3.13 – *Balanced Recall* по класах для моделі з ResNet152v2 (C10img65)

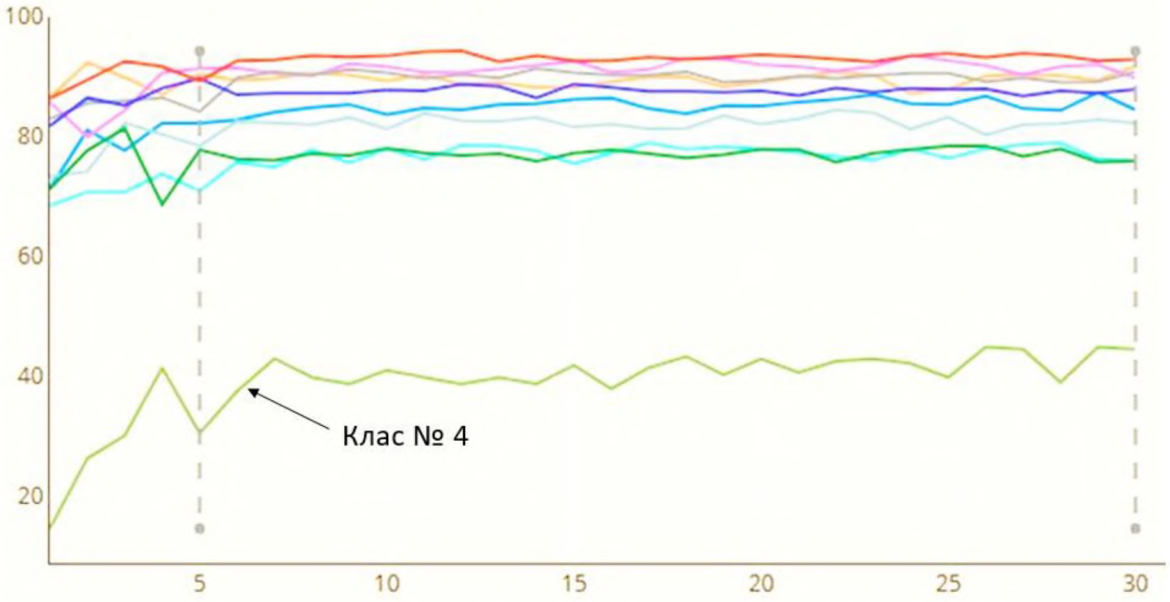

Рисунок 3.14 – *Balanced Recall* по класах для моделі з DensNet169 (C10img65)

В умовах введеного дисбалансу за параметром *Balanced Recall* і з точку зору вимог до обчислювальних потужностей найкраще себе зарекомендувала модель з DensNet169. Як наслідок, надалі доцільно дослідити пріоритети вибору архітектури CNN для класифікації зображень.

3.2 Рекомендації щодо використання моделі глибокого навчання згорткових нейронних мереж класифікації зображень

Для реалізації моделі глибокого навчання Transfer Learning необхідно визначитись з вибором архітектури нейронної мережі. Вибір між ResNet та DenseNet залежить від конкретного завдання, обсягу даних та обчислювальних ресурсів. На основі проведених досліджень можливо сформувати деякі рекомендації, які можуть допомогти прийняти рішення.

Архітектура ResNet вважається пріоритетною для таких випадків.

1. Великі датсети. ResNet зазвичай добре працює з великими наборами даних, такими як ImageNet. Якщо у вас є великий обсяг даних, ResNet може дати хороші результати.

2. Transfer Learning. Передбачені моделі ResNet доступні для багатьох популярних фреймворків глибокого навчання, що робить їх зручними для перенесення навчання.

3. Обчислювальна ефективність. Незважаючи на глибину, ResNet зазвичай досить ефективні завдяки структурі залишкових сполук. Вони також добре масштабуються у глибині.

На даний час, існує кілька мобільних додатків для класифікації зображень на основі ResNet:

1. AI Image Classification on iOS with ResNet. Ця програма дозволяє класифікувати зображення на основі ResNet на пристроях iOS.

2. Image classification guide for Android. Цей посібник допоможе створити програму для класифікації зображень на основі ResNet для пристроїв Android.

3. ResNet Implementation for Image Classification: Це рішення Kaggle дозволяє класифікувати зображення на основі ResNet.

В ході досліджень встановлено, що архітектура DenseNet є більш доцільною для наступних ситуацій.

1. Менші датасети. DenseNet може бути більш стійкою до перенавчання на менших датасетах через свою щільну структуру, яка служить якоюсь регуляризацією.

2. Ефективне використання параметрів. Якщо єс обмежені обчислювальні ресурси або необхідна модель з меншою кількістю параметрів при збереженні високої точності.

3. Додатки з деталізацією ознак. Щільні сполуки забезпечують більш багате та деталізоване уявлення ознак. Це може бути корисним у завданнях, де важливі деталі, наприклад, у медичних зображеннях.

4. Обчислювальна інтенсивність. Треба мати на увазі, що, незважаючи на ефективне використання параметрів, DenseNet може бути більш вимогливим до ресурсів через щільні з'єднання, особливо на великих датасетах або при великій глибині мережі.

Існує кілька мобільних додатків для класифікації зображень на основі архітектури DenseNet.

1. Image classification guide for Android. Цей посібник допоможе створити програму для класифікації зображень на основі DenseNet для пристроїв Android.

2. New Lightweight DenseNet Based on Mix-Structure Convolution. Дана робота описує модель DenseNet, яка використовує Mix-Structure Convolution.

3. Multipath feature recalibration DenseNet for image classification. Дана робота описує модель MFR-DenseNet для класифікації зображень.

4. Dense-MobileNet Models описана в додатку A Novel Image Classification Approach.

Однак, остаточне рішення бажано приймати після експериментів на конкретному наборі даних, щоб визначити, яка архітектура дає кращі результати. У багатьох випадках різниця у продуктивності між цими архітектурами може бути несуттєвою, і вибір може залежати від інших факторів, таких як час навчання, розмір моделі та ін.

3.3 Збільшення обсягу датасету на основі аугментації

Геометрична трансформація зображень є популярним методом аугментації даних, який дозволяє збільшити обсяг та різноманітність тренувального набору без необхідності збирання нових даних. Згідно п. 1.3, розглянемо докладніше кожну зі згаданих трансформацій.

1. Повороти. Під час цієї трансформації зображення повертається на певний кут. Зазвичай, використовуються невеликі кути (наприклад, в діапазоні від -30 до +30 град.), щоб симулювати нормальні зміни перспективи. Крім того, під час повороту частина зображення може вийти за межі рамки, і з'являються порожні області. Їх, зазвичай, заповнюють, застосовуючи різні стратегії, наприклад, відображенням чи константним значенням.

2. Зсув. Зображення може бути зміщене по горизонталі або вертикалі на певну кількість пікселів або відсотків від вихідного розміру. Як і при поворотах, зсув може спричинити появу порожніх областей на зображенні, яке потрібно заповнити.

3. Масштабування. В даному випадку, зображення збільшується або зменшується. Це може бути корисним, щоб симулювати зміни у відстані між об'єктом і камерою. Масштабування може бути рівномірним (однакова зміна за шириною та висотою) або нерівномірним, де зміна розмірів по одній з осей буде більшою, ніж за іншою.

4. Відображення зображення по горизонталі або вертикалі створює дзеркальне відображення. Горизонтальне відображення особливо корисне для зображень, де орієнтація об'єкта не має значення (наприклад, фотографії тварин або автомобілів). Однак, варто бути обережним у випадках, коли орієнтація має значення (наприклад, текст на зображенні, тримання предметів у руці людини та ін.).

При застосуванні геометричних трансформацій важливо враховувати їх комбінації. Наприклад, поворот та зсув можуть бути застосовані одночасно до

одного зображення, що збільшує кількість можливих варіантів аугментації. Також варто зазначити, що застосування аугментацій має бути релевантним завданням. Наприклад, у завданнях медичної діагностики надто агресивна аугментація може зробити зображення нереалістичними і таким чином менш корисними для навчання. Також варто враховувати, що аугментація даних має використовуватись обережно, щоб вона відповідала конкретному випадку використання та типу даних, щоб не вводити модель в оману.

Більшість бібліотек глибокого навчання, таких як TensorFlow [44] або PyTorch [45], мають інструменти для простого застосування цих аугментацій. Згідно завдання, в роботі основний акцент робиться на фреймворк TensorFlow. Він має кілька інструментів та методів для аугментації зображень. Серед них слід виділити наступні. Основний модуль TensorFlow, який надає множину рандомних функцій для маніпулювання зображеннями є tf.image, наприклад:

1. Зміна розміру та обрізка:

– resize() – змінює розмір зображення до вказаних розмірів;

– resize_with_pad() – змінює розмір зображення зі збереженням співвідношення сторін, додаючи відступи, якщо це необхідно;

– central_crop() – обрізає центральну частину зображення;

– random_crop() – випадково обрізає частину зображення до вказаного розміру.

2. Аугментація:

– random_flip_left_right() – випадкове горизонтальне відображення зображення;

– random_flip_up_down() – випадкове вертикальне відображення зображення;

– random_brightness() – випадкова зміна яскравості зображення;

– random_contrast() – випадкова зміна контрастності зображення;

– random_hue() – випадкова зміна відтінку зображення;

– random_saturation() – випадкова зміна насиченості зображення;

– random_jpeg_quality() – випадкова зміна якості JPEG під час кодування зображення.

3. Аджастменти:

– adjust_brightness() – змінює яскравість зображення на вказане значення;

– adjust_contrast() – змінює контрастність зображення на вказане значення;

– adjust_gamma() – змінює гаму зображення на вказане значення;

– adjust_hue() – змінює відтінок зображення на вказане значення;

– adjust_saturation() – змінює насиченість зображення на вказане значення.

4. Перетворення кольорів:

– rgb_to_grayscale() – перетворює зображення з формату RGB на відтінки сірого;

– grayscale_to_rgb() – перетворює зображення із відтінків сірого на формат RGB;

– rgb_to_yuv() – перетворює зображення з RGB на YUV;

– yuv_to_rgb() – перетворює зображення з YUV на RGB.

5. Інші операції:

– transpose() – транспонує зображення;

– rot90() – повертає зображення на 90 градв.

Модуль tf.data.Dataset. Якщо є набір даних у форматі tf.data.Dataset можна легко застосовувати аугментації до зображень за допомогою методу map:

```
def augment(image, label): image = tf.image.random_flip_left_right(image) image =
tf.image.random_flip_up_down(image) return image, label dataset = dataset.map(augment)
```

У TensorFlow модуль tf.keras.layers.experimental.preprocessing включає набір шарів попередньої обробки, які можна використовувати в моделях Keras для виконання аугментації на льоту під час навчання. Деякі з цих шарів включають:

– RandomFlip – випадкове відображення зображення у вертикальному або горизонтальному напрямку;

– RandomRotation – випадковий поворот зображення;

– RandomZoom – випадкове масштабування зображення;

– RandomTranslation – випадкове зміщення зображення.

Результат залучення інструментарію TensorFlow для виконання процедур аугментації наведено в Додатку А. В якості робочого середовища використано хмарний сервіс Google Colaboratory.

3.4 Техніко-економічне обґрунтування прийнятих рішень

Запропонований в роботі підхід щодо врахування впливу дисбалансу датасету є досить актуальним в завданнях обробки тексту, наприклад, сегментація тексту на зображенні [46, 47], OCR [48] та ін.

Розробка коду Python з використанням ImageDataGenerator є економічно виправданою і може принести значні вигоди компанії, особливо якщо головною метою є покращення якості моделей глибокого навчання. Згідно [49], фахівець-фрілансер «Python developer» в середньому в Україні заробляє 44000 грн на місяць. Вважатимемо, що за одну години програміст здатен розробити 20 рядків коду. На код Python програми ImageDataGenerator довжиною 900 рядків він витратить 45 годин. Відповідно, йому треба заплатити 2475 грн. Але крім цього треба враховувати ще час на тестування коду (≈ 15 годин). Таким чином, оплата послуг даного фахівці складає 3300 грн. ImageDataGenerator використовує інструментарій відкритої бібліотеки Keras, тому ліцензійні витрати відсутні.

В свою чергу, вартість написання коду Python для Transfer Learning з використанням архітектури DensNet може сильно варіюватися в залежності від низки наступних факторів. Для визначення обсягу проекту – чи потрібно лише імплементувати передачу навчання з DensNet, або ж потрібно розробляти додатковий код для обробки даних, візуалізації результатів,

тестування моделі тощо? Досвід розробника – розробник з більшим досвідом у галузі глибокого навчання може вимагати вищу оплату, але при цьому він може виконувати роботу швидше та ефективніше. Географія – вартість послуг розробників може відрізнятися в залежності від країни або регіону. Терміни – у випадках, коли проект потрібно завершити якнайшвидше, можуть бути додаткові витрати. Якщо припустити, що наймається розробник з середнім рівнем кваліфікації в Україні, то вартість за одну годину складає ≈ 950-1550 грн. Час, необхідний для розробки Transfer Learning з архітектурою DensNet може зайняти від 8 до 24 годин в залежності від складності завдання і додаткових вимог. Таким чином, з врахуванням стану ринку, приблизна оцінка вартості проекту може варіюватися від 7600 до 37200 грн. При використанні певного програмного забезпечення можуть виникнути додаткові витрати. Однак, Python є відкритим та безкоштовним, тому в даному випадку ліцензійні витрати відсутні. На тестування та внесення змін потрібно ще 8-10 годин роботи. Таким чином, загальні витрати на послуги з розробки програмного коду складають ≈ 47200 грн.

Висновки до розділу 3

Для оцінки впливу дисбалансу на точність класифікації нейронної мережі використано Transfer Learning VGG16, ResNet152v2 і DensNet169. Розглядалось два датасети (без дисбалансу та з дисбалансом 65 %). Відмінності в архітектурі CNN проявились в результатах досліджень. Найбільш стійкою до впливу дисбалансу виявилась модель глибокого навчання на базі DensNet169.

Враховуючі специфіку завдань класифікації зображень, в роботі обгрунтовано рекомендації щодо використання моделі глибокого навчання CNN в контексті вибору архітектури та формування якісного датасету.

З метою підвищення продуктивності запропонованих варіантів моделей глибокого навчання в роботі розглянутий програмний код Python, що спирається на інструментарій ImageDataGenerator бібліотеки Keras. Це дозволяє не тільки збагатити датасет, але і надає можливість компенсувати наявний дисбаланс між класами.

При застосуванні Transfer Learning для завдань класифікації зображень потрібно звертати увагу на трансформацію розмірів вхідних зображень до параметрів використаної архітектури CNN.

Для підтвердження можливості практичної реалізації запропонованих підходів виконано техніко-економічне обґрунтування прийнятих рішень. Наприклад, витрати за послуги з розробки програмного коду складають приблизно 40000 грн.

ВИСНОВКИ

Для вивчення того, як незбалансованість даних впливає на ефективність роботи нейронних мереж, створено декілька наборів даних, базуючись на CIFAR-10, що має 10 категорій. Ці набори мають різний ступінь нерівномірності розподілу даних між класами. Дисбаланс має певні характеристики та параметри. Обрання розміру навчальних та тестових даних, а також максимальний рівень незбалансованості відбувається так, щоб модель нейронної мережі для класифікації зображень була ефективною.

Для класифікації зображень було обрано певну структуру нейронної мережі. Основна увага була приділена згортковим нейронним мережам, таким як VGG, ResNet та DenseNet, які були детально вивчені під час наукових досліджень.

У зв'язку з присутністю дисбалансу класів, стандартні метрики оцінювання роботи мережі можуть бути неефективними для моделей глибокого навчання, що класифікують зображення, оскільки вони можуть впливати на якість та стабільність моделі. Додатково, у певних задачах важливо одночасно правильно визначати позитивні випадки (забезпечуючи високу чутливість) і знижувати число помилок (гарантуючи високу точність). Тому було розглянуто використання метрик, які враховують ці особливості, або окреме оцінювання продуктивності для кожної категорії.

Основна увага приділяється Balanced Recall, як заміні традиційній чутливості, особливо у випадках, коли дані мають нерівномерне розподілення класів. Цей показник визначається як середнє значення Recall (або точності) для всіх класів. При використанні вагових коефіцієнтів слід пам'ятати, що їх введення може утруднити розуміння результатів. Тому необхідно детально аналізувати вплив цих коефіцієнтів на ефективність моделі.

У дослідженні було запропоновано застосувати Transfer Learning, аби дослідити вплив нерівноваги в датасетах. Цей метод дозволяє ефективніше обробляти дані, економити час на навчання, досягати кращої генералізації,

зменшувати необхідний об'єм початкових даних та розв'язувати конкретні завдання. В процесі дослідження були розглянуті ключові етапи інтеграції переднавчених CNN в модель глибокого навчання для класифікації зображень.

Щоб оцінити, як дисбаланс датасету впливає на точність класифікації, були використані моделі Transfer Learning, такі як VGG16, ResNet152v2 та DensNet169. Два розглядуваних датасети відрізнялись ступенем дисбалансу, один із них мав дисбаланс на рівні 65%. Вплив різних архітектур CNN був видний у результатах: модель на основі DensNet169 показала найкращу стабільність до дисбалансу.

У вивченні завдань класифікації зображень розглядалися рекомендації для використання моделі глибокого навчання CNN у контексті вибору її структури та створення ефективного датасету. Для покращення роботи моделей глибокого навчання було вивчено код на Python, який використовує інструменти ImageDataGenerator з бібліотеки Keras. Це допомагає розширити датасет і вирівняти розподіл даних між класами. При використанні Transfer Learning у задачах розпізнавання зображень важливо адаптувати розміри вхідних зображень до вимог обраної архітектури CNN.

Щоб підтвердити практичну втілення запропонованих методів, було здійснено технічний та економічний аналіз прийнятих рішень. Зокрема, витрати на послуги програмування становлять близько 40000 грн.

Таким чином, результатами роботи є модель глибокого навчання згорткових нейронних мереж класифікації зображень; рекомендації щодо використання моделі глибокого навчання згорткових нейронних мереж класифікації зображень. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні хмарних сервісів.