Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МОДЕЛЕЙ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ БІБЛІОТЕКИ KERAS ДЛЯ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ НА ЗОБРАЖЕННІ З ВИКОРИСТАННЯМ МЕТОДІВ ІНТЕРПРЕТАЦІЇ РІШЕНЬ МОДЕЛІ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-19-1
Новічонок М.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
(код і повна назва)
Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Новічонок Марії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей згорткових нейронних бібліотеки Keras для класифікації об'єктів на зображенні з використанням методів інтерпретації рішень моделі

затверджена наказом по університету від «23» жовтня 2020 року № 1428Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 25 листопада 2020 р.
3. Вихідні дані до роботи: Методи класифікації об'єктів на зображенні, методі інтерпретації рішень моделей, бібліотека Keras, мова програмування Python, згорткові нейронні мережі, метод Inception V3, модель MobileNet, модель ResNet, алгоритм LIME, міри якості нейронних мереж
4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд класичних методів класифікації об'єктів на зображенні
2. Вимоги методів машинного навчання
3. Вирішення задачі класифікації за допомогою нейронних мереж
4. Огляд існуючих методів інтерпретації рішень моделі
5. Метод LIME
6. Аналіз загальної структури згорткових нейронних мереж
7. Аналіз структури та характерних особливостей мережі Inception V3
8. Аналіз структури, характерних особливостей мережі MobileNet V1
9. Аналіз структури, характерних особливостей мережі ResNet
10. Порівняння попередньо навчених моделей Keras Applications
11.Створення застосунку для класифікації об'єктів на зображенні за допомогою моделей Keras Applications та мови Python
12. Використання бібліотеки LIME для інтерпетації рішень моделей

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Приклади задач комп'ютерного зору, приклади інтерпетацій рішень моделі, схеми нейронних мереж, ілюстрація роботи програми, порівняльний аналіз моделей Keras Applications, фрагмент анкети для проведення експертного оцінювання

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на до атестаційної роботи	23.10.2020	
2	Аналіз завдання, підбір літератури	03.09.20-10.09.20	
3	Огляд технологічних засобів та математичних методів вирішення задачі класифікації	11.09.20-03.10.20	
4	Огляд існуючих методів інтерпретації рішень моделі	04.10.20-20.10.20	
5	Програмна реалізація	20.10.20-02.11.20	
6	Проведення дослідження обраних моделей	02.11.20-07.11.20	
7	Проведення експертного оцінювання	08.11.20	
8	Оформлення пояснювальної записки	08.11.20-18.11.20	
9	Перевірка на плагіат	19.11.20	
10	Рецензування	20.11.20	
11	Підготовка презентації та доповіді	23.11.20	
12	Занесення роботи в електронний архів	29.11.20	
13	Попередній захист атестаційної роботи	30.11.20	

Дата видачі завдання 23 10 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Яковлева О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка атестаційної роботи: 78 с., 8 табл., 38 рис., 2 додатки, 40 джерел.

КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, НАВЧАННЯ З УЧИТЕЛЕМ, ІНТЕРПРЕТАЦІЯ, DATASET IMAGENET, БІБЛІОТЕКА KERAS, MOBILENET, RESNET, LIME.

Робота присвячена вирішенню проблеми класифікації об’єктів на зображенні з використанням моделей Keras Applications та методів інтерпретації отриманих рішень. Розглянуто існуючі класичні та сучасні підходи та методи класифікації об’єктів на зображенні. Проаналізовано модель згорткової нейронної мережі. Досліджено різновиди згорткових нейронних мереж, їх структуру та характерні особливості. Розглянуто методи інтепретації рішень моделей, що використовуються для роботи з зображеннями. Виконано порівняння точності моделей. Проведено експертну оцінку щодо зрозумілості отриманих інтерпретацій результатів. Обрано моделі нейронної мережі з найвищою оцінкою експертів та найбільшою точністю класифікації об’єктів на зображенні.

Спираючись на результати досліджень було розроблено застосунок для класифікації об’єктів на зображенні на основі попередньо навчених моделей бібліотеки Keras та мови Python. Для інтерпретації результатів використано методи бібліотеки LIME.

IMAGE CLASSIFICATION, MACHINE LEARNING, NEURAL NETWORK, TEACHER LEARNING, INTERPRETATION, DATASET IMAGENET, KERAS LIBRARY, MOBILENET, RESNET, LIME.

Thesis was devoted to a solution of the problem of the objects classification on the image with use of Keras Applications and methods of interpretation of the obtained results. The existing classical and modern approaches and methods of classification of objects on the image ware considered. The model of a convolutional neural network was investigated. The variety of convolutional neural networks, their structure and characteristic features was investigated. Methods of interpretation of decisions of the models used for work with images were considered. The comparisons of accuracy of the models is executed. An expert assessment of the clarity of the obtained interpretations of the results was performed. The neural network models with the highest expert score and the highest accuracy of object classification in the image was chosen.

The application was based on a conducted research. The developed application use previously trained Keras API models and the Python language. LIME library methods were used to interpret the results.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface
BBox – Bounding Box
CAM – Class Activation Mapping
CNN – Convolutional Neural Network;
DL – Deep Leaning
DAG – Directed Acyclic Graph
GAP – Global Avegare Pooling
GPU – Graphics Processing Unit
KNN – K-Nearest Neighbors
LIME – Local Interpretable Model-agnostic Explanations
ML – Machine Learning;
ReLU – Rectified Linear Unit
RMSProp – Root Mean Square Propagation
RNN – Reccurent Neural Network
SVM – Support Vector Machine

ВСТУП

Дана робота присвячена дослідженню питання класифікації об’єктів на зображеннях та проблемі інтерпретації отриманих результатів. Актуальність дослідження підтверджується розповсюдженням методів машинного навчання (Machine Learning (ML)) у більшості сфер нашого життя. Нейронні мережі ідентифікують, чи справді людина, а не робот, намагається отримати доступ к сайту. Базуючись на методах комп’ютерного зору розроблено десятки програм, що полегшують роботу медиків при визначенні захварювання пацієнта (більш того – впорюються іноді краще за самого спеціаліста) [1]. Не менш актуальним є питання інтепретації рішень моделей, адже відповідальність за результати, надані програмою, та складність задач підвищилась в декілька разів.

Для виконання поставленої задачі атестаційної роботи було розглянуто класичні та новітні методи вирішення задачі класифікації об’єктів на зображеннях. Проаналізовано питання структури згорткових мереж. Для проведення дослідження та програмної реалізації було проаналізовано можливості мови Python та бібліотеки Keras для роботи з моделями нейронних мереж. Надано увагу методам інтерпретації моделей, їх класифікації та особливостям використання. Обрано алгоритм LIME (Local Interpretable Model-agnostic Explanations, агностичні локальні інтепретовані пояснення моделі) для проведення експертної оцінки якості моделей бібліотеки Keras.

У результаті передатестаційної практики було створено застосунок для класифікації об’єктів на зображеннях з використанням декількох моделей згорткових нейронних мереж, бібліотеки Keras та мови Python. Роботу моделей було порівняно за розрахованою точністю пошуку об’єктів, а також за оцінками експертів щодо зрозумілості отриманої інтерпретації рішень кожної моделі.

1 СУЧАСНИЙ СТАН ПИТАННЯ ВИРІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ НА ЗОБРАЖЕННІ

1.1 Успіхи у вирішенні задач комп'ютерного зору

Усі задачі комп'ютерного зору полягають у імітації бачення та сприймання людиною оточуючих її об'єктів. Проте на відміну від людей, для яких процеси детектування, класифікації, порівняння та ін. операцій із зображеннями є фактично однією моментальною підсвідомою операцією, то для комп'ютера на кожну з операцій є свій складний алгоритм. Існує близько 10 окремих задач комп'ютерного зору [2]:

1. Детектування об'єктів (Object Detection).
2. Класифікація зображень (Image Classification).
3. Виявлення візуальних стосунків (Visual Relationship Detection).
4. Створення описів до зображень (Image Captioning).
5. Реконструкція зображень (Image Reconstruction).
6. Розпізнавання обличь (Face Recognition).
7. Сегментація екземпляру (Instance Segmentation).
8. Семантична сегментація (Semantic Segmentation).

Детектування об'єктів – це здатність правильно виявляти або ідентифікувати об'єкти на зображенні та області, у якій вони знаходяться, у вигляді прямокутників (відомих як обмежувальні рамки (Bounding Boxes (BBoxes)).

Задача класифікації зображень, що будет разглянута далі у даній роботі, полягає у визначенні, до якого класу належить кожен об'єкт, присутній на вхідному екземплярі, або класу, що характерезує усе зображення в цілому (наприклад, якщо відбувається класифікація сцени – природного ландшафту чи типу приміщення).

Виявлення візуальних стосунків є більш складним завданням, що націлено не тільки на класифікацію об'єктів, але і на пошук взаємозв'язків між

ними. Фактично, такі методи ідентифікують процеси, що відбуваються на екземплярах. Зазвичай, вихід алгоритму є триплетом, що поєднує суб’єкт, дію та об’єкт (наприклад, < «людина», «грає», «фортепіано»> .

Image Captioning – це задача створення пояснень (або субтитрів) до зображення. На відміну від Visual Relationship Detection, такі моделі генерують не окремі слова, а повноцінні речення, що можуть містити детальний опис того, що відбувається (наприклад «Дівчинка у червоній сукні грає з лялькою»).

Реконструкція зображення полягає у відновленні частин екземпляру, що були втрачені або зіпсовані (наприклад, у випадках виконання зйомки за низького освітлення, що призводить до великої кількості шуму). Прикладом є дослідницький проект Nvidia, результати роботи якого зображені на рис 1.1.

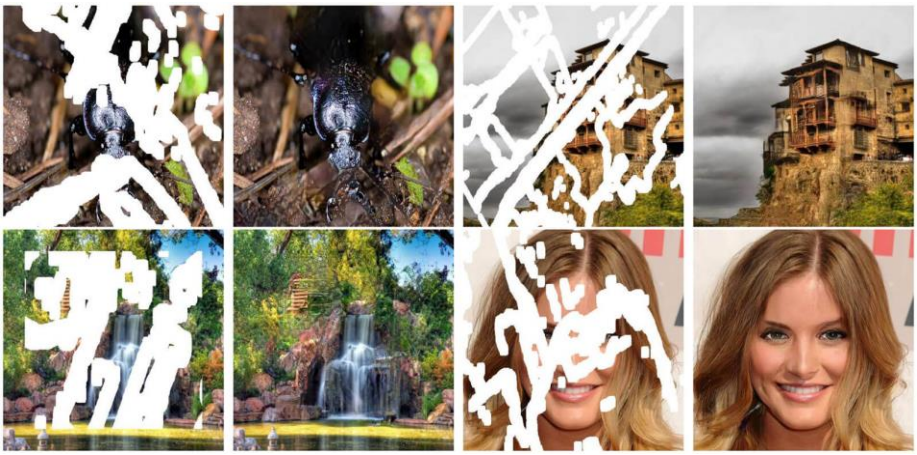

Рисунок 1.1 – Приклад реконструкції зображення

Також окремо розглядається задача Face Recognition, що має в собі одразу 3 підзадачі: детекція обличчя (face detection), процес захоплення обличчя (face capture) та процес зіставлення (face match). Отже, для розпізнавання облич необхідно власне знайти його на зображенні, представити перелік рис людини у вигляді цифрової інформації та зіставити з іншим фото, щоб визначити, чи належать вони одній людині [3].

Задача сегментації полягає у знаходженні об’єкта та виділенні усіх пікселів, що до нього відносяться. Для Instance Segmentation важливо зазначити, де знаходиться межа між пікселами одного об’єкта та інших. Якщо

кажуть про задачу семантичної сегментації, то такі алгоритми намагаються виділити об’єкти, що відносяться до одного класу (зазвичай, за допомогою кольору).

Приклади вирішення задач комп’ютерного зору представлені на рисунку 1.2.

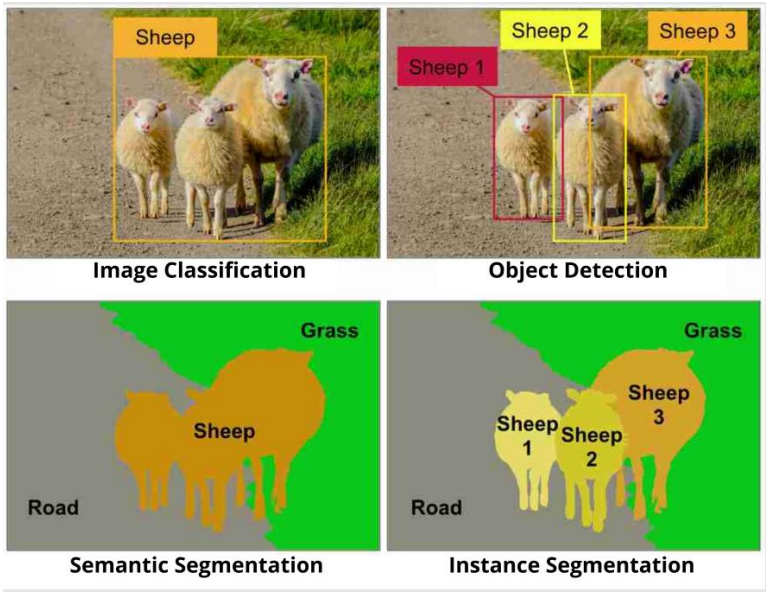

Рисунок 1.2 – Приклади вирішення задач комп’ютерного зору

1.2 Огляд підходів та методів класифікації об’єктів на зображенні

Задача класифікації – це задача розбиття множини об’єктів або спостережень на апріорно задані групи, названі класами, всередині кожної з яких вони вважаються схожими один на одного та мають приблизно однакові властивості й ознаки. При цьому рішення здійснюється на основі аналізу значень атрибутів (ознак) [4]. Кількість класів завжди обмежена і чітко означена (у випадку, коли кількість класів невідома, кажуть про задачу кластеризації). Класифікація за двома класами називається бінарною.

Існує близько 10 різних методів класифікації. У випадку, коли вхідними даними є зображення, найчастіше використовують наступні: [5]:

– метод k-найближчих сусідів;

– метод опорних векторів;
– дерева рішень;
– нейронні мережі.

1.2.1 Класичні підходи

Метод k-найближчих сусідів (Nearest Neighbors (KNN)) відноситься до метричних методів і вважається найпростішим класифікатором. Об’єкт відноситься до того класу, який є найбільш поширеним серед сусідів даного елементу. Перевагами даного методу є проста реалізація, пророблена теоретична база, адаптація під потрібне завдання вибором метрики або ядра, простота інтерпретації. До недоліків відносяться:

– недостатня продуктивність в реальних завданнях, так як число сусідів, які використовуються для класифікації, буде досить великим;

– складність в наборі відповідних ваг і визначенням, які ознаки необхідні для класифікації;

– залежність від обраної метрики відстані між прикладами.

Метод опорних векторів (support vector machines (SVM)) – це набір алгоритмів, що використовуються для задач класифікації та регресійного аналізу. З огляду на те, що в N-вимірному просторі кожен об’єкт належить одному з двох класів, SVM генерує гіперплощину розмірності ($N-1$) з метою поділу цих точок на 2 групи (рис. 1.3). Крім того, що метод виконує сепарацію об’єктів, SVM підбирає гіперплощину таким чином, щоб та характеризувалася максимальним віддаленням від найближчого елемента кожної з груп.

Серед найбільш масштабних проблем, які були вирішені за допомогою методу опорних векторів (і його модифікованих реалізацій) виділяють розпізнавання статі на підставі фотографії та сплайсинг людської ДНК.

Потенційні недоліки методу опорних векторів полягають у неможливості калібрування ймовірності попадання в певний клас та

складності інтерпретації параметрів моделі. Також важливо зазначити, що метод SVM може бути використаний лише для бінарної класифікації. Але незважаючи на дані недоліки, метод опорних векторів вважається одним з найкращих для вирішення задачі класифікації.

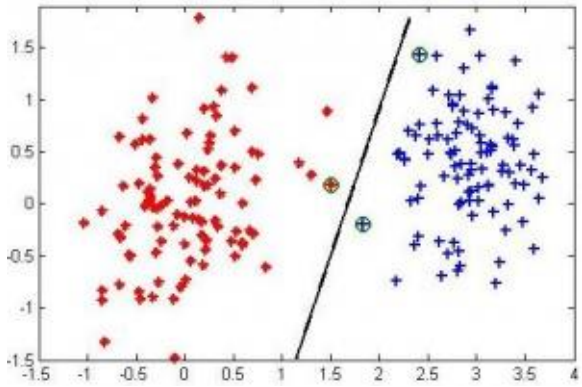

Рисунок 1.3 – Візуалізація методу SVM

Дерево прийняття рішень – засіб підтримки прийняття рішень, яке використовує деревовидний граф або модель прийняття рішень, а також можливі наслідки їх роботи, включаючи ймовірність настання події, витрати ресурсів і корисність. Кожен вузол дерева містить умову розгалуження за однією з ознак. У кожного вузла стільки розгалужень, скільки значень має обрана характеристика.

Головною перевагою методу є висока продуктивність навчання і прогнозування, такі дерева рішень можна легко візуалізувати і інтерпретувати.

1.2.2 Основні підходи машинного навчання та вимоги до них

Власне поняття «машинне навчання» приймає досить багато значень у різній літературі. Найбільш влучним та водночас простим визначенням можна вважати наступне:

Машинне навчання – методи штучного інтелекту, що стають точніше та краще зі зростанням кількості повторів їх запуску.

Слід сказати, що саме ця особливість – здатність до навчання – відрізняє алгоритми машинного навчання від усіх інших методів, для яких навпаки важливо зберігати одноманітність результату роботи алгоритму за будь-якої кількості його використання на одних і тих самих даних. Чим краще модель навчена – тим точніші результати її роботи.

Якість усіх алгоритмів машинного навчання залежить від двох факторів: від структури і оптимізації параметрів побудованої моделі та від даних, на яких вона навчалася. На практиці, саме якість набору даних найбільше впливає на результати.

Зауважимо, що завдання побудови самої моделі не менш складне, ніж пошук або створення датасету, який би відповідав поставленій задачі. Найчастіше, недоліками даних, які негативно впливають на навчання моделі є відсутність структурованості даних, «білі плями» та зайві дані. Під «білими плямами» слід розуміти відсутність частини важливих для аналізу даних. Прикладом білої плями є пропущенні значення віку пасажирів у датасеті Titanic [6]. Зайві дані не впливають на результуюче значення, але перенавантажують вибірку (наприклад, порядкові номери записів у таблиці).

Також до розповсюджених проблем можна віднести помилкові значення (від'ємне значення суми кредиту) та відсутність одноманітності (представлення числових даних у вигляді слів та цифр для різних рядків таблиці). У випадку, коли дані містять завелику кількість помилкових значень або серед характеристик об'єкту важко знайти закономірності (має місце слабка кореляція даних), виникає underfitting. Даним терміном називають випадок, коли помилка на тестових даних значно перевищую помилку на тренувальних даних. Також ситуація недостатнього навчання моделі може виникнути у випадку помилкового вибору методу виявлення характерних ознак.

Отже, перед тим як подавати дані на вхід моделі, необхідно позбутися усіх недоліків та знайти найважливіші «фічі» вибірки. Фічею у машинному навчанні називають інформаційну ознаку кожного елементу вибірки, яка

впливає на результуюче значення. Для покращення датасету необхідно виконати наступні кроки:

1. Позбутися компонентів, які є постійними або мають низьку варіативність, а отже, не є статистично вагомими.

2. Серед двох (або більше) показників, між якими існує пряма лінійна залежність, обрати лише один. Інші лінійно залежні показники є зайвими і впливають на статистику так само, як і обрана фіча.

3. Провести feature selection (відбір фіч) на основі розрахунку feature importance (вагомості фіч) [7].

Feature selection дозволяє відфільтрувати усі показники даних та знайти найбільш вагомі для поставленої задачі. Також цей процес дозволяє позбутися тих характеристик, які слабо корелюють з результуючим значенням та можуть призвести до перенавчання моделі (overfitting). Результатом відбіру є набір ознак з найкращими параметрами моделі на тренувальному датасеті [8].

Зауважимо, що видалення даних не є обов’язково умовою створення якісного набору даних. Сучасні методи навчання моделей дозволяють обробляти моделі великої розмірності, проте існує пряма залежність між розмірністю та часом обробки даних – чим більше ознак тренувального датасету, тим довше триває навчання моделі.

1.2.3 Вирішення задачі класифікації на основі нейронних мереж

Нейронні мережі є найсучаснішим методом серед перерахованих вище методів. Їх ефективність досить висока, тому що вони генерують фактично велике число регресійних моделей (які використовуються в рішенні задач класифікації статистичними методами).

Дуже довгий час з моменту створення перших нейронних мереж їх все ще вважали інструментом, що складно використовувати для більшості задач машинного навчання. Найголовнішою причиною був час, що витрачався на

тренування моделей – процес займав тижні та навіть місяці. Особливо це стосувалось мереж для обробки зображень, що вимагали значно більшого об’єму пам’яті. Проте у 2012 році розробником Олексієм Крижевським було створено нейронну мережу AlexNet, що розпочала «революцію» згорткових нейронних мереж (рис. 1.4).

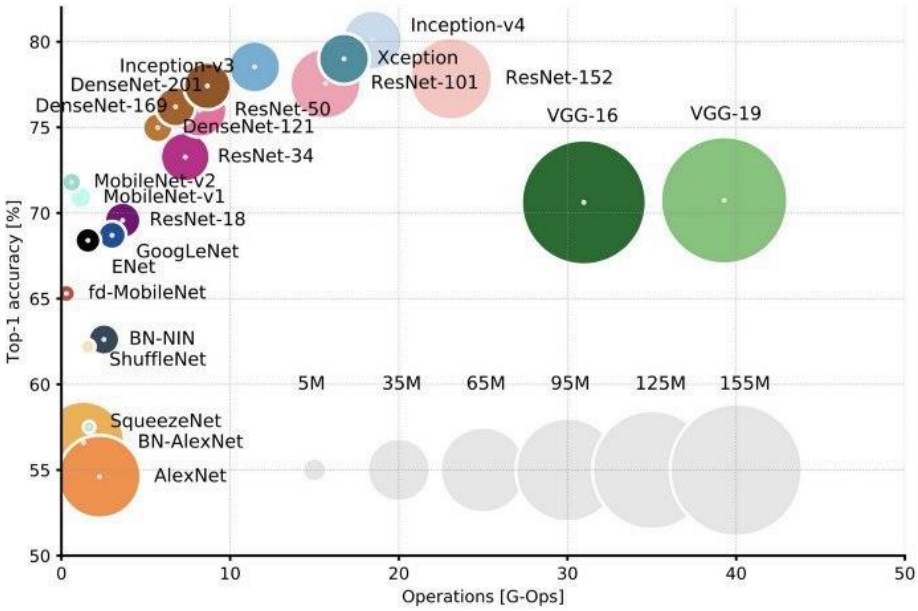

Рисунок 1.4 – Графік точності існуючих мереж відносно кількості операцій

Характерними особливостями мережі AlexNet є використання функції активації ReLU (Rectified Linear Unit), додавання шару виключення (dropout) та шару максимального пулінгу (max-pooling). Також створена у той час нова відеокарта NVIDIA, що була використана для навчання моделі, дозволило у 10 разів зменшити час на тренування мережі. AlexNet стала доказом того, що нейронні мережі можливо використовувати не тільки у рамках наукових досліджень, але і для реальних завдань бізнесу [9].

Проте слід пам’ятати, що будь-який метод, заснований на нейронних мережах, ніколи не дасть класифікатор потрібної якості, якщо набір прикладів не буде достатньо повним для навчання мережі.

Детальніше про архітектуру згорткових нейронних мереж описано у розділі 2.

1.2.4 Використання навчених моделей

Задача класифікації зображень дуже часто вирішується за допомогою трансферного навчання (transfer learning), що дозволяє значно зменшити час на створення нової мережі. Для переносу навчання використовують pre-trained (попередньо навчені) моделі, які вже були натреновані на великому наборі тестових зображень, подібних до тих, що необхідно класифікувати у новій задачі. Найбільш популярними pre-trained моделями є згорткові нейронні мережі з архітектурами VGG16, MobileNet, Inception тощо. Вони реалізовані у таких популярних бібліотеках як Tensorflow та Keras і можуть бути легко додані до проекту.

Для того, щоб перенавчити модель під власні потреби, необхідно видалити оригінальний класифікатор, додати власний та завершити налаштування (fine-tune) вже нової мережі, користуючись однією з трьох стратегій (рис. 1.5):

1. Виконати тренування усієї моделі, використовуючи власний набір даних. За таких умов обов’язково необхідно мати великий датасет для досягнення високої точності нейронної мережі.

2. Часткове «заморожування» шарів. Під час вибору, які шари залишити замороженими (під час тренування значення коефіцієнтів таких шарів залишиться незмінним), слід орієнтуватися на розмір власного датасету та кількість параметрів мережі.

У випадку, коли кількість параметрів велика, а датасет містить лише кілька тисяч примірників, навчають лише декілька шарів, щоб запобігти явищу перенавчання. Якщо датасет налічує десятки тисяч тестових зображень, а кількість параметрів є малою, можливо покращити модель, тренуючи більше шарів для конкретної задачі. Отже, даний метод вимагає від розробника пошуку балансу між кількістю параметрів мережі, розміром датасету та кількістю шарів, що залишаться незмінними під час налаштування.

3. «Заморожування» згорткової основи (convolutional base). Фактично, під час такого навчання модель використовується як механізм вилучення ознак для подальшого подання їх до класифікатора. Така методика підходить у таких випадках, коли:

– модель вирішує задачу дуже подібну до вимог нової задачі;

– обчислювальна потужність є недостатньою для навчання усієї моделі;

– невеликий набір даних для тренування [10].

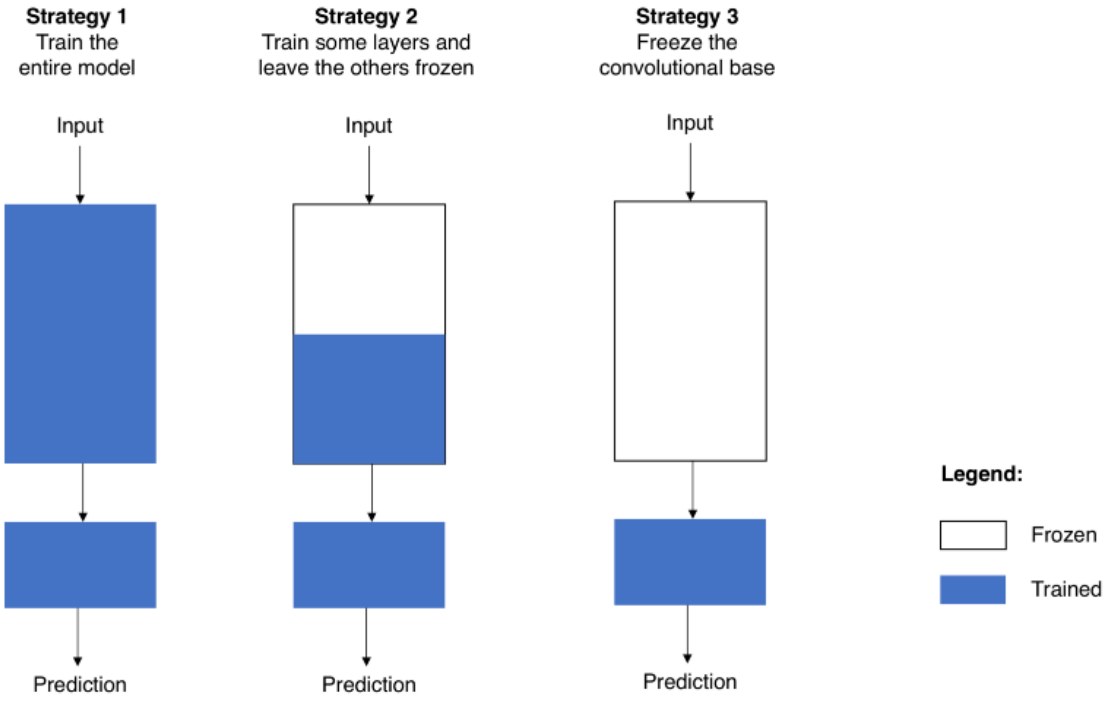

Рисунок 1.5 – Візуалізація стратегій fine-tune

Під час використання першої або другої стратегії необхідно звернути увагу на такий гіперпараметр мережі як коефіцієнт швидкості тренування (learning rate), що показує, наскільки необхідно корегувати ваги мережі під час навчання. Якщо обрати занадто високе значення коефіцієнту, зростає ймовірність спотворення якості моделі.

1.3 Питання інтерпретації рішень щодо класифікації

Зазвичай, під час оцінки якості моделі за міру приймаються такі показники як точність та/або швидкість. Такий підхід досі є загальноприйнятим під час проведення змагань (наприклад, Kaggle) або розробці проектів. Проте останні кілька років вчені та розробники підіймають питання інтерпретації моделей, а саме їх структури та результатів. Також кажуть про випадки, коли для покращення показників моделі недостатньо лише оперувати її параметрами та якістю даних, що використовуються для навчання нейронної мережі.

Власне поняття інтерпретації щодо методів машинного навчання не має чіткого визначення та пов’язане з поняттям зрозумілості. Інтерпретацією моделі вважається такий процес, що надає нову додаткову інформацію, зрозумілу для людини та достатню для подальшого вирішення завдань, повязаних з даною моделлю [11]. Слід зазначити, що зрозумілість та корисність отриманої інтерпретації не є однаковою для кожного реціпієнта. Повнота та вид наданої інтерпретації повинні бути різними для розробника моделі та користувача системи, у якій задіяно алгоритми машинного навчання. Щодо класифікації надамо приклад, пов’язаний з класифікацією захворювання на рентгенівських знімках. Для розробника більш вагомою інформацією буде інтепретація впливу параметрів моделі на отриманий результат. Для експертної групи медиків – виділення тих областей зображення, що дійсно відповідають захворюванню, яке спрогнозувала програма.

Отже, одним з головних завдань сьогодні є перехід від понять «black box» («чорний ящик»), яким часто називають процеси у нейронній мережі, та «output» («вихідне значення»). Натомість приходять поняття «explainable model» («інтерпретуєма модель») та «explanation interface» («інтерфейс інтепретації»). Перехід від старої моделі роботи з алгоритмами до нової проілюстровано на рисунку 1.6.

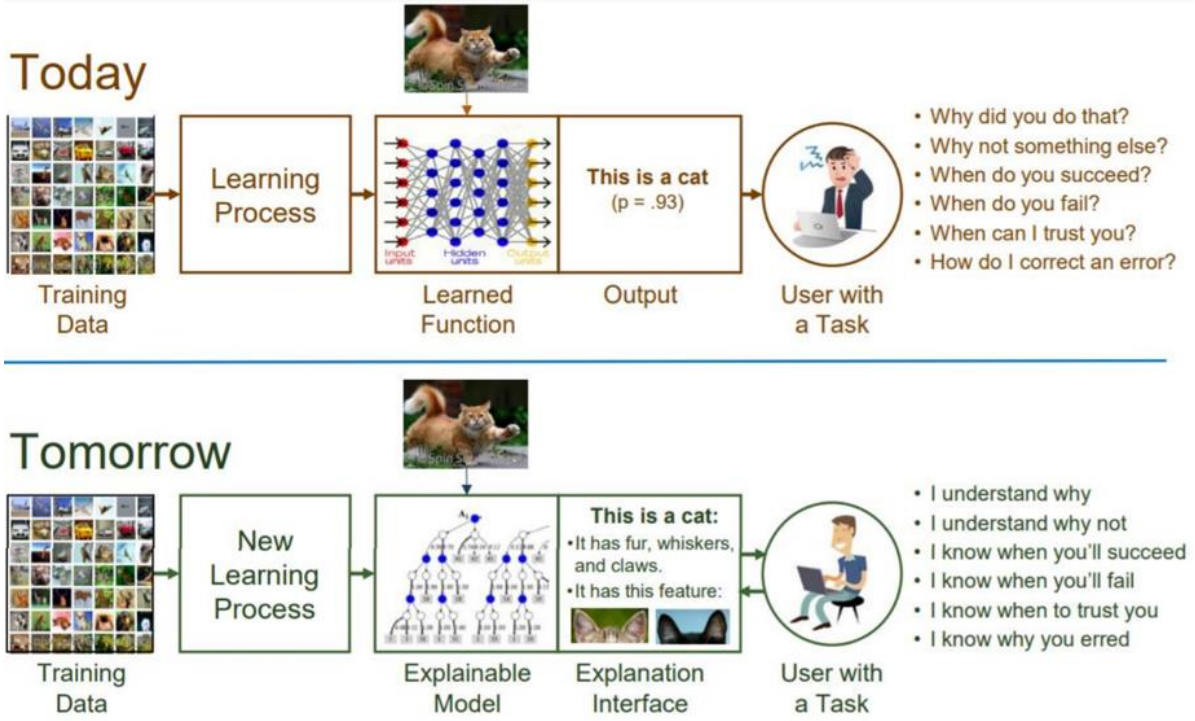

Рисунок 1.6 – Схематичне зображення вимог до інтепретації

1.3.1 Необхідність пояснення рішень класифікаторів

Machine Learning все частіше обирається за основну технологію при розробці проектів у сферах медицини, фінансів, що вимагають підвищенної відповідальності та зведення ймовірності помилок до нуля [12]. Саме тому виникає необхідність у довірі до моделі та впевненості у її роботі.

Існують так звані змагальні атаки (adversarial attacks), розроблені саме для перевірки якості моделі щодо правильності її відповідідей – чи справді вони засновані на характерних особливостях об’єкта, а не на випадковості [13]. Для виконання такої атаки використовують спеціальні змагальні приклади (adversarial examples) – зображення, що були певним чином «зіпсовані» (наприклад, накладанням шуму, як на рисунку 1.7). Також прикладом таких зображень є фотографії об’єктів, що дуже схожі між собою, але відносяться до різних класів (рис. 1.8).

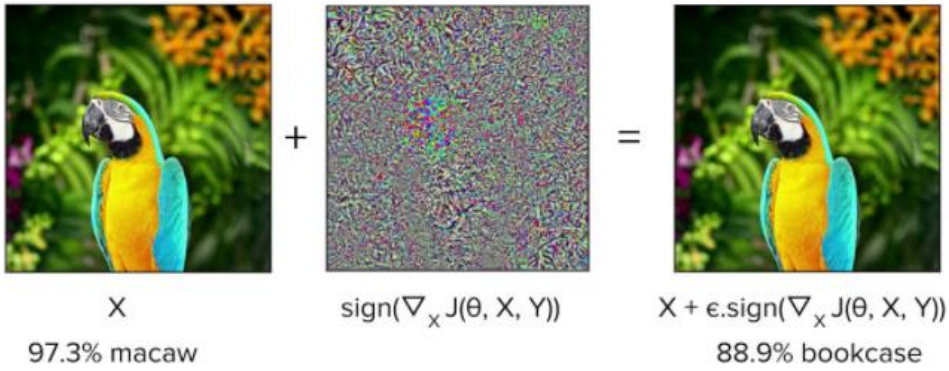

Рисунок 1.7 – Результат змагальної атаки на зображення за допомогою накладання шуму

Існує багато методів для створення змагальних прикладів. Більшість підходів пропонують мінімізувати відстань між змагальним прикладом та екземпляром, яким слід маніпулювати, одночасно зсунувши передбачення на бажаний (змагальний) результат. Деякі методи вимагають доступу до градієнтів моделі, що, звичайно, працює лише з градієнтними моделями, такими як нейронні мережі, інші методи вимагають доступу лише до функції прогнозування, що робить ці методи модельно-агностичними. Під модельно-агностичними методами розуміють ті методи, що є універсальніми для будь-якої моделі машинного навчання [14].

Рисунок 1.8 – Приклад adversarial examples

1.3.2 Огляд існуючих підходів до інтерпретації рішень моделей нейронних мереж

Для інтерпретації нейронних мереж необхідно використовувати спеціальні методи інтерпретації. Вони позбавляють від необхідності власноруч оцінювати мільйони математичних рішень, що були зроблені моделлю від надходження зображення на вхід мережі і до отримання передбачення.

Методи інтепретації мають наступну класифікацію:

– глобальні та локальні;

– залежні та незалежні від моделі;

– методи інтепретаціїї до побудови, під час побудови та після побудови моделі (або інтепретація результатів).

Глобальні методи використовуються для пояснення моделі загалом на усій множині входів, у той час як локальні методи створені для інтепретації конкретних входу та виходу мережі. Вважається, що локальні методи є кориснішими, адже допомогають знайти конкретну область, що призвела до помилкових результатів.

Залежні та незалежні від моделі методи вже були згадані вище. До незалежних відносять агностичні моделі. Прикладом залежного методу для нейронних мереж є використання механізму Attention (надання певним атрибутам об’єкта більшої важливості під час навчання мережі) [15].

Найбільшу увагу у даній роботі надано поясненню рішень моделі, а саме візуалізація тих областей зображення, що найбільше вплинули на вихід нейронной мережі. Прикладами таких методів є:

– маски чутливості (Saliency Maps);

– алгоритм LIME (Local Interpretable Model-agnostic explanations, локальньні інтепретовані пояснення агностичної моделі);

– картографування активації класу (Class Activation Mapping (CAM)) та градієнтно-зважене картографування активації класу (Grad-CAM).

Маски чутливості є класичним методом інтерпретації та полягають у виділенні тих частин зображення, що викликали найбільшу активність шарів моделі. Зазвичай, Saliency Maps являють собою теплові карти (рис. 1.9).

Рисунок 1.9 – Приклад маски чутливості

CAM та Grad-CAM є більш сучасною версією масок чутливості. Механізм отримання теплової карти полягає у використанні шару глобального середнього об’єднання (Global Avegare Pooling (GAP)) замість останнього повнозв’язного шару мережі. Важливість регіонів визначається шляхом передання вагів GAP шару до шару softmax (на зразок механізму зворотнього поширення помилки) [16, 17].

Алгоритм LIME використовується для інтерпретації класифікаторів, що на вхід приймають зображення, текст або табличні дані. Оскільки метод є агностичним, він сприймає класифікатор як «чорний ящик». В основі алгоритму покладено виконання «збурення» (perturbation) вхідного екземпляру шляхом приховування частини зображення [18]. Більш детально про алгоритм LIME викладено у розділі 2.

Також існує так звана «інтерпретація чорного ящика», що дуже схожа на інтепретацію рішень моделі, але дозволяє відстежити виходи не тільки останнього шару, але й окремого нейрону або шару. Для такої інтепретації розроблена бібліотека Lucid від Google [19]. Приклад візуалізації нейронів мережі під час прийняття рішення зображено на рисунку 1.10.

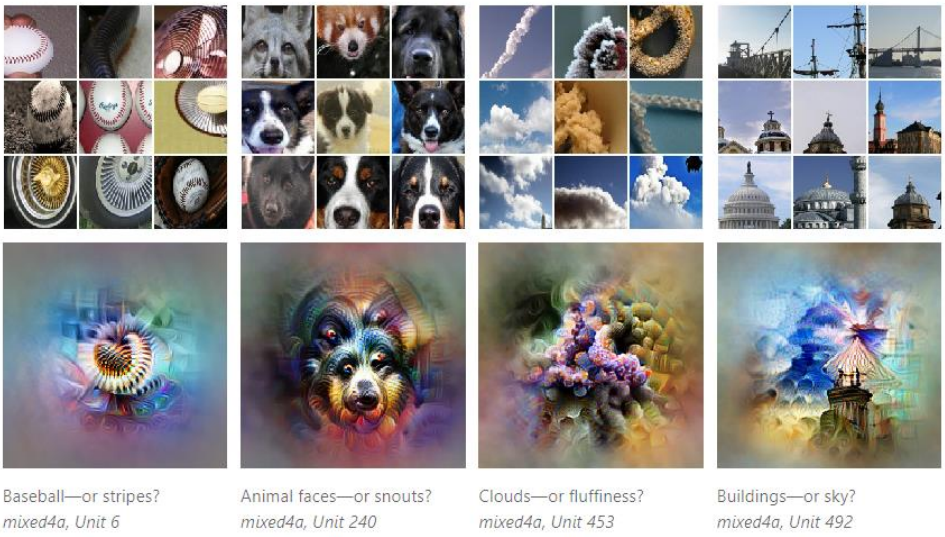

Рисунок 1.10 – Візуалізація нейронів мережі за допомогою бібліотеки Lucid

1.4 Програмні аспекти вирішення задач машинного навчання

За останні два десятиліття була створена велика кількість фреймворків, що дозволяють вирішувати задачі машинного навчання. Більшість з них є у відкритому доступі, можуть використовуватись як для досліджень, так і для розробки комерційних проектів. Бібліотеки для ML проектуються таким чином, щоб розробник мав змогу вирішувати поставлену задачу, не зупиняючись на найскладніших математичних та структурних аспектах моделей, а використовував та розширював вже готові рішення фреймворків.

Задачі машинного навчання мають наступні вимоги:

1. Підтримка мов програмування з високою швидкістю обробки програмного коду, зрозумілим інтерфейсом і повноцінним та максимально повним спектром функцій для математичних обчислень, оскільки усі методи Machine Learning базуються на математичній статистиці та математичному аналізі.

2. Наявність достатньої кількості документації та тренувальних матеріалів (офіційна документація, проекти на GitHub, запити на StackOverflow).

3. Моделювання згорткових та рекурентних нейронних мереж (Convolutional Neural Networks (CNN), Reccurent Neural Network (RNN), використовуючи готові модулі.

4. Обраний функціонал повинен мати гарну графічну базу для відображення результатів роботи моделі та її проміжних станів. Бажано, щоб бібліотека надавала можливість легко зображувати різноманітні графіки, графи та області значень.

5. Підтримка графічного процесору (Graphics Processing Unit (GPU)) для прискорення обчислень.

Серед популярних мов програмування, що сумісні з багатьма бібліотеками ML та відповідають переліченим вимогам є мова Python. У свою чергу найбільш популярним фрейморком з такими характеристиками можна назвати TensorFlow.

1.4.1 Мова Python, її модулі та бібліотеки для вирішення задач машинного навчання

Мова Python є інтерпретованою мовою з високою швидкістю виконання програмного коду, що є досить суттєвим для задач машинного навчання. Як вже було сказано, Python займає перші позиції серед мов, що використовуються у галузі ML. Дана мова має усі необхідні модулі для реалізації математичної моделі задачі, для графічного зображення її результатів та, найголовніше, навчання.

До бібліотек першої необхідності мови Python для Machine Learning відносяться наступні:

– Scikit-Learn – інструмент з відкритим вихідним кодом, що має велику швидкість обробки даних та найчастіше використовується для вирішення задач регресії, класифікації, обрання моделі та кластеризації. Є необхідним у сферах Data Mining та Data Analysis;

– Pattenr – безкоштовний модуль, що дозволяє збирати та обробляти дані, аналізувати якість мереж та має зручний функціонал для візуалізації. Також має велику кількість вбудованих універсальних тест-кейсів.

1.4.2 Огляд існуючих бібліотек для роботи з нейронними мережами

Виділимо найбільш застосовувані бібліотеки машинного навчання. До них відносяться:

– Keras – зручна надбудова (обгортка) над бібліотеками більш низького рівня (наприклад, TensorFlow та Theano). Являє собою програмний інтерфейс застосунку (Application Programming Interface (API)) високого рівня для навчання глибоких нейронних мереж;

– TensorFlow – це бібліотека з відкритим вихідним кодом для досліджень та розробки програмних продуктів. TensorFlow пропонує API для початківців і експертів для розробки для настільних, мобільних, веб-застосунків та хмарних сховищ [20];

– PyTorch – сучасний фреймфорк для навчання глибоких нейронних мереж та побудови графів обчислень (статичний та динамічний). Не схожа на інші бібліотеки машинного навчання та має більшу ймовірність написання «чистого коду» через зменшення кількості задання обчислень.

1.4.3 Google середовище для машинного навчання Colaboratory

Як вже було сказано, найважливішими програмними аспектами у навчанні моделей машинного навчання є швидкість обробки коду та підтримка бібліотек, які націлені на розробку у сфері ML. Якщо раніше для цього був потрібен комп’ютер з графічним процесором та налаштоване середовище для виконання програмного коду, то сьогодні достатньо мати профіль у Google.

Colaboratory – це безкоштовне хмарне середовище для створення та розробки Jupyter-ноутбуків на мові Python. Основна перевага цього сервісу – можливість підключення GPU Tesla K80, потужного графічного процесору. Також до плюсів Colaboratory можна віднести:

– підтримка версій Python 2.7 та 3.6;

– просте завантаження фреймворків;

– завантаження та збереження файлів з GitHub, Google Drive, Sheets та Google Cloud Storage;

– створення інтерактивних форм та віджетів (списків, графіків, анімацій тощо);

– створення моделей TensorFlow та підтримка TensorBoard;

– завантаження наборів даних за допомогою Pandas.

Отже, у даному розділі було розглянуто задачу класифікації та сучасний стан її вирішення. Описані сучасні програмні засоби вирішення задач машинного навчання в цілому та задачі класифікації об’єктів на зображеннях. Було приділено увагу необхідності інтерпретації результатів роботи алгоритмів ML та існуючим методам дослідження внутрішньої структури моделей та їх виходів.

1.5 Постановка задачі дослідження

Актуальність даного дослідження полягає у необхідності перевірки якості існуючих моделей нейронних мереж та інтепретованості їх результатів класифікації для користувача.

Об’єктом дослідження є визначення найбільш значущих ознак зображення, обраних нейронною мережею для класифікації об’єкта.

Метою дослідження є створення застосунку для класифікації об’єктів на зображенні з використанням моделей Keras Applications та методів

інтерпретацій рішень моделей. Для виконання поставленої задачі необхідно розглянути такі теоретичні питання:

– вивчити структуру згорткових мереж;

– дослідити згорткові нейронні мережі MobileNet та ResNet, їх покоління, характерні особливості структури;

– дослідити питання інтепретації рішень нейронних мереж;

– дослідити в порівняльному аспекті якість класифікації нейронними мережами Inception V3, MobileNet V1, ResNet-101 V2, ResNet-152 V2 за допомогою метрик precision та recall;

– провести дослідження щодо інтепретації вищеназваних нейронних мереж за допомогою бібліотеки LIME;

– провести експертне оцінювання інтепретацій та проранжувати отримані результати методом середніх арифметичних рангів та методом медіан.

Ознайомитися з такими практичними аспектами:

– моделі бібліотеки Keras, попередньо навчені для класифікації об’єктів на зображенні;

– мова програмування Python, її модулі та бібліотеки для роботи з нейронними мережами та візуалізацією даних;

– бібліотека LIME, її функції та трактування отримуваних результатів.

2 ДОСЛІДЖЕННЯ МОДЕЛЕЙ БІБЛІОТЕКИ KERAS З ВИКОРИСТАННЯМ МЕТОДІВ ІНТЕРПРЕТАЦІЇ ТА РОЗРОБКА АЛГОРИТМІВ ДЛЯ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ

2.1 Структура згорткових нейроних мереж для вирішення задачі класифікації

Найчастіше для вирішення задачі класифікації за наявності велокої кількості класів використовують згорткові нейронні мережі. Архітектура мережі обирається таким чином, щоб закласти апріорні знання про предметну область:

– локальна кореляція – кожен піксель зображення пов'язаний з сусіднім пікселем;

– інваріантність до зміщення – об'єкт пошуку може зустрітися у будь-якій області зображення.

Для того, щоб пояснити мережі важливість зв'язку між сусідніми пікселями зображення, до неї додають локально-зв'язаний шар або згортковий шар. Відмінність локальних шарів від повноз'єднаних у тому, що на вхід до цього шару подається лише локальна частина зображення. Локально-зв'язаний шар має однакові ваги у різних частинах вхідного зображення. Інваріантність до зміщення також досягається шляхом додавання зготкового шару. Важливою задачею є обрати оптимальний розмір фільтрів даного шару та їх кількість для обробки всього зображення.

Отже, типова структура CNN складається зі згорткового шару, пулінгу та повноз'єднаного шару. Прикладом простої згорткової мережі є одна з перших мереж такого типу LeNet (рис. 2.1). Архітектура LeNet складається з вхідних даних, згорткового шару, активаційної функції, шару пулінгу та повноз'єднаного шару.

Вхідний шар складається з власне зображень, які будуть проаналізовані нейронною мережею. У різній літературі вхід або вважають першим шаром,

або називають просто вхідними даними мережі. Вхід складається з початкової інформації про зображення: його розміри та канали кольорів [21].

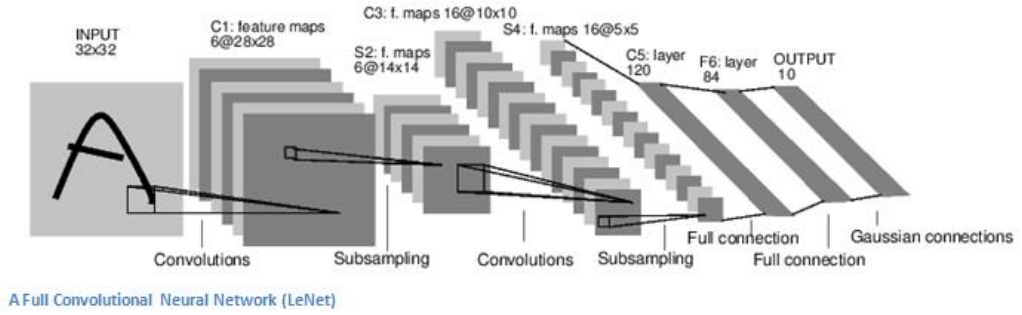

Рисунок 2.1 – Згорткова нейронна мережа LeNet

Згортковий (convolutional) шар дозволяє об’єднувати значення розташованих поруч пікселів і виділяти більше узагальнені ознаки зображення. Для цього по картинці послідовно ковзають квадратним вікном невеликого розміру (3×3, 5×5, 7×7 пікселів тощо), що називається ядром (kernel). Кожен елемент ядра має свій ваговий коефіцієнт, що перемножується зі значенням того пікселя зображення, на який в даний момент накладено елемент ядра. Потім отримані для всього вікна числа складаються, і ця зважена сума дає значення чергової ознаки об’єкту.

Для отримання матриці («карти») ознак всього зображення, ядро послідовно зсувається по горизонталі і вертикалі. У наступних шарах операція згортки застосовується вже до карт ознак, отриманим з попередніх шарів.

Зображення або карти ознак у межах одного шару можуть скануватися не одним, а кількома незалежними фільтрами, даючи таким чином на вихід не одну карту, а кілька. Такі карти іноді називають «каналами». Налаштування ваг кожного фільтра відбувається за допомогою все тієї ж процедури backpropagation.

Якщо ядро фільтра при скануванні не виходить за межі зображення, розмірність карти ознак буде менше, ніж у вихідної картинки.

Для спрощення математичного опису згорткових нейронних мереж будемо вважати, що вхідні зображення мають однакову висоту та ширину. Отже, операція двовимірної згортки матриці X розміром $X \times X$ та ядром K розміром $k \times k$ описується наступним виразом:

$$B_{ij} = (X \times K)_{ij} = \sum_{\alpha=0}^{k-1} \sum_{\beta=0}^{k-1} X_{i+\alpha, j+\beta} K_{\alpha\beta,} \, 0 \le i, j < x - k + 1 . \qquad (2.1)$$

Якщо потрібно зберегти той же розмір, застосовують так звані paddings – значення, якими доповнюється зображення по краях, і які потім захоплюються фільтром разом з реальними пікселями картинки. Окрім paddings на зміну розмірності так само впливають strides – значення кроку, з яким вікно переміщається по зображенню (карті). Приклад застосування фільтрів з додаванням додаткових пікселів наведено на рисунку 2.2.

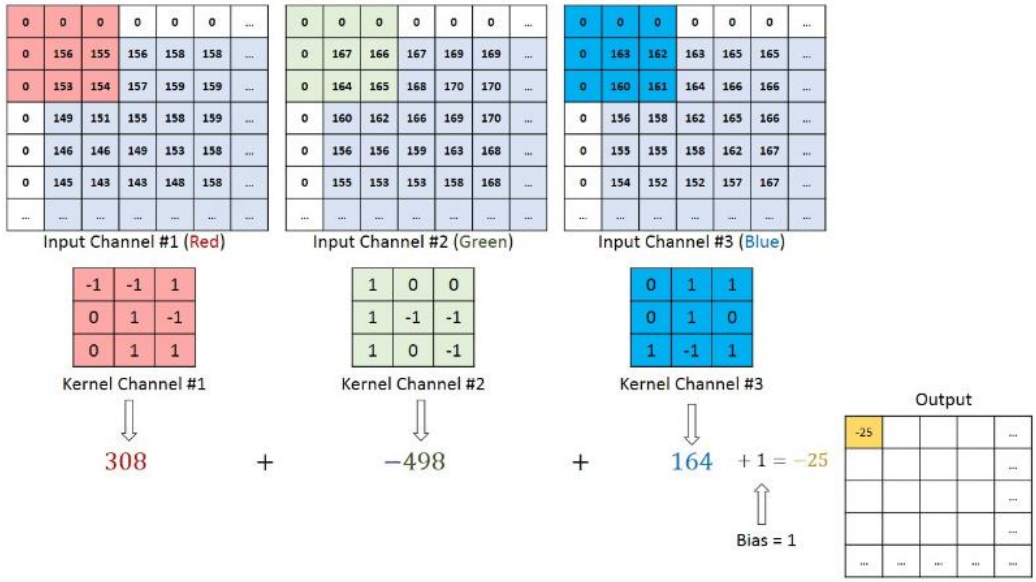

Рисунок 2.2 – Застосування згортки до карти зображення

Після шару згортки завжди йде шар пулінгу для зменшення чутливості до просторового зсуву [22]. Пулінг зменшує розмірність зображення у p разів. Коли pooling виконується з максимальним значенням параметру p, кажуть

про поняття «макс-пулінгу». Математичним записом пошуку максимального значення p є:

$$C_{ij} = \max_{\alpha,\beta=0\ldots p-1} \left\{ B_{pi+\alpha, pj+\beta} \right\}, 0 \le i, j < n/p\,. \tag{2.2}$$

Після операції макс-пулінгу до зображення застосовується зміщення b_{ij} (порогове значення нейрону) та деяка функція активації f:

$$D_{ij} = f\,(C_{ij} + b_{ij}). \tag{2.3}$$

Зміщення b_{ij} задається випадково для кожного нейрону до початку навчання та є незмінним.

Прикладом активаційної функції може бути функція ReLU. Вона є однією з найпопулярніших функцій активації завдяки великій кількості її модифікацій під різні задачі та простоті задання

$$\mathrm{ReLU} = \max(x, 0). \tag{2.4}$$

Останній шар зазвичай є повноз’єднаним та має у собі функцію softmax. Функцію активації м’якого максимуму зручно застосовувати для задач класифікації, тому що вона дозволяє трактувати вихідні значення нейронів як ймовірність приналежності до певного класу, а також забезпечує, щоб тільки одне вихідне значення було близьке до одиниці за рахунок застосування експоненти [23]. За умови, коли шар містить у собі n нейронів, функція softmax має вигляд:

$$y_j = \mathrm{softmax}(z_j) = \frac{\exp(z_j)}{\sum_{i=1}^{n} \exp(z_j)}, \tag{2.5}$$

де $z_i = \sum_k w_{ki} xk + b_i, x_k$ – входження k-го нейрону;

w_{ki} – матриця вагів;

b_i – зміщення.

Отже, для CNN можна виділити наступні особливості:

1. У найпростішому випадку архітектура CNN – це набір шарів, які перетворюють зображення у результат роботи мережі (визначення класу).

2. Кожен шар відповідає за певний етап процесу обробки зображення (шар згортки, активації, пулінг і повноз'єднаний шар). Дана архітектура є базовою і має певні модифікації.

3. Кожен шар отримує на вході об'ємну 3D інформацію і трансформує зі збереженням 3D-об'єму за допомогою функції, що диференціюється.

4. Шар може не мати параметрів (шаг згортки та повноз'єднаний шар – мають, ReLU і pooling – не мають).

5. Шар може не мати додаткові гіперпараметри (наприклад, шаг згортки, повноз'єднаний шар та пулінг мають, функція активації ReLU – не має).

2.2 Архітектури та моделі нейронних мереж, що досліждуються

2.2.1 Моделі бібліотеки Keras

Бібліотека Keras є потужним засобом для розробки нейронних мереж. Головна ідея даного API була у швидкому досягненні результатів під час проведення досліджень та експериментів.

Keras надає три основні способи створення нейронних мереж:

1. Послідовні моделі (Sequential model).

2. Keras Functional API.

3. Підкласові моделі (Model subclassing).

Послідовні моделі є швидким рішенням, але можуть складатися лише з простого набору шарів, кожен з яких має рівно один вхідний і один вихідний

тензор. Тензором може бути як окреме число, вектор ознак з розв’язуваної задачі або зображення, так і цілий набір («батч») описів об’єктів або масив із зображень.

Послідовна модель не підходить, коли:

– модель повинна мати кілька входів або кілька виходів;

– будь-який з шарів повинен мати кілька входів або кілька виходів;

– є необхідність у спільному доступу до шарів;

– потрібна нелінійна топологія (наприклад, залишковий зв’язок (residual connection), розгалужена модель).

Keras Functional API дозволяє створювати більш гнучкі моделі, ніж Sequential. Основна ідея полягає у створенні спрямованого ациклічного графу (directed acyclic graph (DAG)), яким, зазвичай, є типова модель глибокого навчання. Переваги, що надає Keras Functional API, наступні:

– можливість використання спільних шарів;

– модель може мати шари з кількома входами та виходами;

– створений граф можливо використовувати для декількох моделей;

– створення складної архітектури з вкладенням моделей.

Model subclassing рекомендується для використання у випадках, коли надані бібліотекою структури шарів не підходять под нестандартну задачу розробника. У такому випадку він може скористатися функціоналом побудови власних шарів нейронної мережі.

У рамках даної роботи розглядаються Keras Applications – це моделі глибокого навчання (deep learning (DL)), які пропонуються разом із попередньо підготовленими вагами. Ці моделі можна використовувати для прогнозування, вилучення функцій, тонкого налаштування (fine-tuning), а також самостійно, якщо задача не є вузькоспеціалізованою, а націлена, наприклад, на дослідження архітектури моделей та їх порівняння.

Моделі, створені будь-яким з перелічених методів, можуть взаємодіяти між собою в одному проекті [24].

Усі моделі Keras Applications попередньо треновані на датасеті ImageNet. На сьогодні ImageNet є найбільшою колекцією копій оригінальних зображень, що знаходиться у відкритому доступі. Дана бібліотека має більш ніж 14 мільйонів ієрархічно структурованих картинок, об’єкти на яких описані словами за допомогою великої кількості експертів із різних країн [25]. Моделі можуть класифікувати лише один об’єкт на зображенні або визначити клас, до якого відноситься усе зображення (ландшафт або тип приміщення). Детальніше алгоритм роботи з даними моделями наведено у розділі 3.

Далі у розділі надано опис моделей, що були задіяні у експерементальній частині даної роботи: Inception V3, MobileNet V1 та архітектура ResNet. Саме ці моделі досягли найбільшого успіху серед згорткових нейронних мереж [26].

2.2.2 Архітектура Inception

У порівнянні з попередніми існуючими архітектурними рішеннями (наприклад, VGG16), мережі Inception виявилися більш обчислювально ефективними як за кількістю параметрів, що генеруються мережею, так і за економічними витратами (пам’ять та інші ресурси) [27]. Модель є кульмінацією ідей, розроблених багатьма дослідниками, та заснована на роботі: «Переосмислення архітектури Inception для комп’ютерного зору» дослідників К. Сегеді та ін. [28].

Передумовою для розробки архітектури Inception стала проблема у виборі правильних розмірів згортки, що підійдуть як для зображень, на яких об’єкт займає дуже мало місця, так і для примірників, на яких об’єкт є ключовою фігурою та займає усі площу. Малі ядра згортки краще підійдуть для локально розташованих об’єктів, а великі – для глобально розташованої інформації. Наївне закладання великої кількості згорткових шарів значно знижчує обчислювальну ефективність. Також існувала проблема перенавчання моделі та через збільшення кількості шарів для поглиблення

мережі стає неможливим якісне виконання зворотнього розповсюдження помилки.

Отже, для рішення вищеперерахованих задач було виконано наступні кроки:

1. Згортки розташовані на одному рівні для поширення мережі замість поглиблення. Також виконується масимальний пулінг, виходи об’єднуються та передаються на наступний шар.

2. Для зниження обчислювальної складності перед згортками з розмірами 5×5 та 3×3 додаються згортки 1×1, що є «дешевшими». Така ж згортка додається після макс-пулінгу.

Перше версія Inception (також відома як GoogLeNet) складалась з 9 модулів, що наведено на рисунку 2.3.

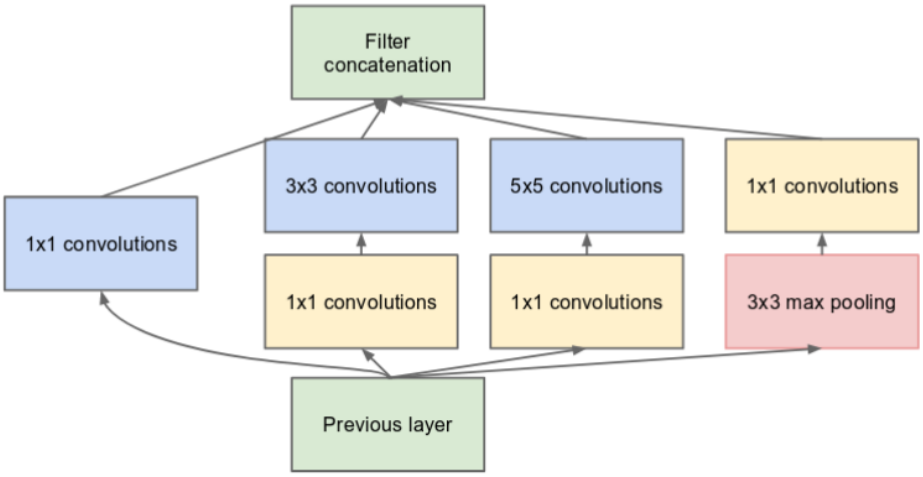

Рисунок 2.3 – Inception модуль зі зниженням розмірності

Для уникнення явища вимивання градієнту, спричиненого глибиною мережі, було додано два допоміжних класифікатора, що використовуються лише під час тренування. Класифікатори є функціями softmax, що застосовуються до виходів двох початкових модулів. У результаті обчислюється загальна функція витрат, що є зваженою сумою основних витрат мережі та допоміжних витрат, обчислених за тими ж мітками.

2.2.2.1 Inception V2

Вдосконалення першої версії архітектури Inception полягали у продовженні збільшення точності моделі та зниження обчислювальної складності. Було вирішено дві проблеми: наявність так званих «репрезентативних вузьких місць» (representatilnal bottleneck) та знижено обчислювальну складність. Під поняттям bottleneck розуміють явище зниження якості моделі через надмірне зменшення розмірів зображення, що призводить до втрати важливої інформації про об’єкт.

Для Inception V2 у архітектуру було внесено такі зміни:

1. Зменшуються розміри ядра згортки для прискорення навчання мережі: замість шарів з розміром 5×5 використовують два послідовні шари з розміром 3×3. Це дозволяє зменшити кількість парамерів зі збереженням розміру входу та глибини виходу.

2. Виконується факторизація (або розкладання) згорткок з розміром $n\times n$ на поєднання згорткок $1\times n$ та $n\times1$. Наприклад, послідовне виконання згорткок 1×3 та 3×1 еквівалентно використання однієї згортки 3×3, проте у такий спосіб обчислювальна вартість знижується на 33%.

3. Для усунення вузьких місць групи фільтрів розташовуються у рамках одного модуля паралельно, а не послідовно, для розширення моделі заміть поглиблення.

Схема модуля Inception V2 зображена на рисунку 2.4.

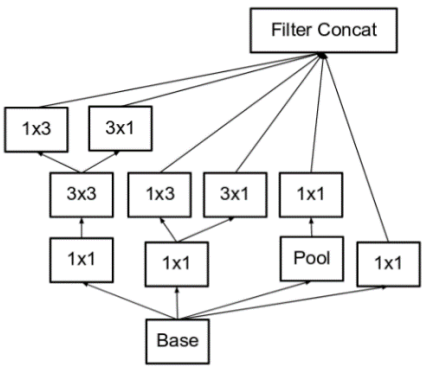

Рисунок 2.4 – Модуль Inception V2

2.2.2.2 Inception V3

Основною метою створення Inception V3 було зниження обчислювальної складності. Автори зазначили, що використання допоміжних класифікаторів не зробило бажаного внеску у роботу моделі через зниження їх ефективності у кінці тренування. Отже, нейрони стають насиченими (значення функції активації стають близькими до 0 або 1), що призводить до вимивання градієнта. Розробка третього покоління мережі була проведена зі збереженням будови основних модулів, що показали високу ефективність у попередній версії. Оновлення Inception V3:

1. Використання оптимізатору RMSProp (Root Mean Square Propagation, (середньоквадратичне поширення)) для градієнтного спуску. Основна ідея оптимізатору полягає у налаштуванні коефіцієнта швидкості тренування для кожного параметру мережі.

2. Факторизація згорткок розміру 7×7.

3. Додається Batch Normalization до допоміжних класифікаторів. Пакетна нормалізація дозволяє використовувати великі значення коефіцієнту швидкості тренування, уникнути проблеми вимивання градієнту та прискорити процес навчання [29].

4. Label smoothing – додатковий регулюючий компонент, що запобігає надмірній «впевненості» мережі у приналежності вхідного зображення до певного класу.

Архітектура мережі Inception V3 представлена на рисунку 2.5

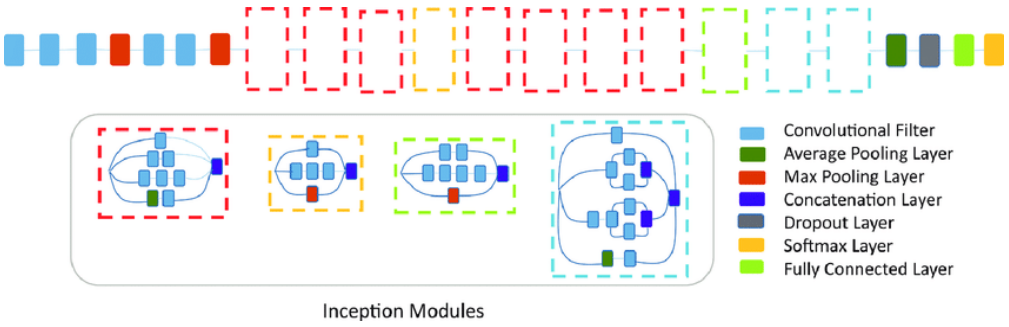

Рисунок 2.5 – Архітектура мережі Inception V3

2.2.3 Архітектура MobileNet V1

Моделі MobileNet відносяться до класу нейронних мереж, що використовуються у сфері мобільних застосунків та вбудованих рішень задач комп'ютерного зору. Основна перевага запропонованої архітектури у швидкості її навчання та оптимізації.

Головний принцип побудови мереж MobileNet полягає у використанні Depthwise Separable Convolution – розділюваної згортки в глибину та у відсутності макс-пулінгу. Для даної архітектури він замінюється на згортковий шар з кроком переміщення ковзного вікна (*stride = 2*)] [30].

Перша версія нейронної мережі (MobileNet V1) складалась з шарів на зразок зображеного на рисунку 2.6.

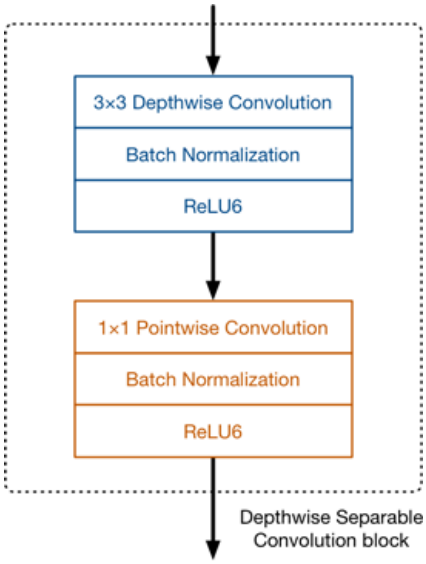

Рисунок 2.6 – Типовий шар мережі MobileNet V1

Повна архітектура MobileNet V1 з 13 шарів розділюваної згортки. Між цими глибинними роздільними блоками немає шарів об'єднання. Замість цього, деякі з шарів глибини мають крок 2 для зменшення просторових розмірів даних. Коли це відбувається, відповідний точковий шар також подвоює кількість вихідних каналів. Якщо вхідне зображення становить $224 \times 224 \times 3$, то вихід мережі становить $7 \times 7 \times 1024$ карти.

Функцією активації, що використовується MobileNet, є ReLU6:

$$\text{ReLU6} = \min(\max(x, 0), 6). \tag{2.6}$$

Останнім шаром зазвичай є або шар softmax, або повноз'єднаний шар. Також особливістю мереж MobileNet є два гіперпараметри: α (множник ширини), що відповідає за кількість каналів кожного шару та ρ (множник глибини), що регулює просторові розміри тензорів [30]. Чим нижче обидва параметри мережі, тим вона швидше та витрачає менший об'єм пам'яті, але водночас нижче якість її результатів.

2.2.3.1 Розділювана згортка

Мережі MobileNet складаються зі звичайної згортки у глибину (на першому шарі) і комбінації depthwise та pointwise (точкової) згорток. Поєднання цих двох принципів і є розділюваною згорткою.

Для типової depthwise згортки характерно застосування фільтру до всіх каналів вхідного зображення. Ковзне вікно переміщується по зображенню і на кожному кроці виконує зважену суму вхідних пікселів, покритих ядром по всіх вхідних каналах.

Важливо, що операція згортки поєднує значення всіх вхідних каналів. Якщо зображення має три вхідних канали, то запуск одного ядра згортки на цьому зображенні призводить до виведення зображення лише з одним каналом на піксель. Таким чином, для кожного вхідного пікселя, незалежно від того, скільки каналів він має, згортка записує новий вихідний піксель тільки з одним каналом [31]. Основна мета depthwise полягає у фільтрації вхідних каналів. На рисунку 2.7 наведено стандартну операцію згортки у глибину.

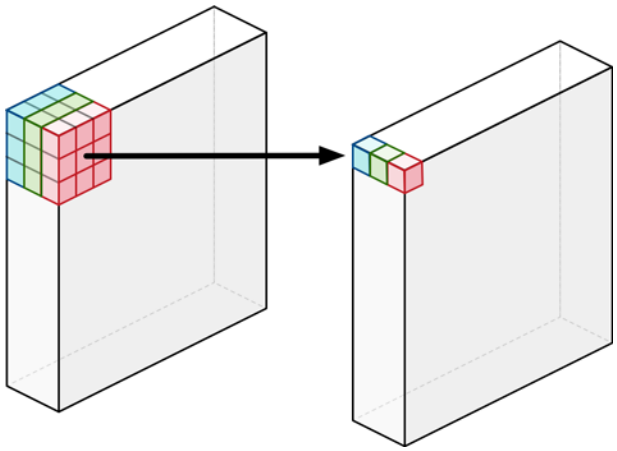

Рисунок 2.7 – Операція згортки у глибину

Інші згорткові шари базуються на точковій згортці (рис. 2.8). На зображення накладається фільтр розмірністю 1×1 та виконується згортка усіх каналів з визначенням їх зваженої суми. Мета точкової згортки полягає в об’єднанні вихідних каналів depthwise згортки для створення нових ознак.

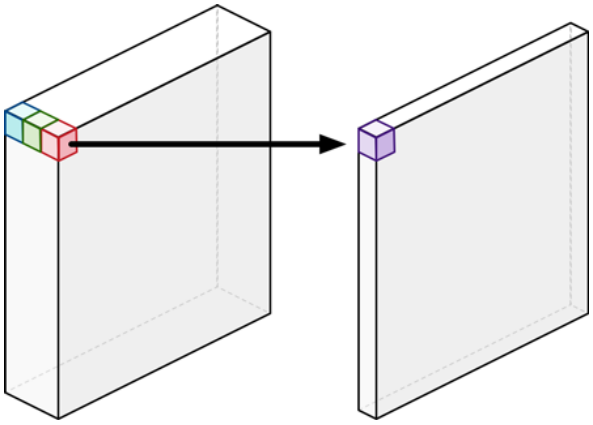

Рисунок 2.8 – Операція точкової згортки

Слід зазначити, що результат роботи звичайної згортки та розділюваної згортки один і той самий – обидва підходи дозволяють отримати об’єднання накладення фільтрів. Проте під час застосування розділюваної згортки обидва етапи виконуються одночасно та з меншою кількістю обчислень і ваг, а отже, з більшою швидкістю. Схематично об’єднання глибокої та точкової згортки зображено на рисунку 2.9.

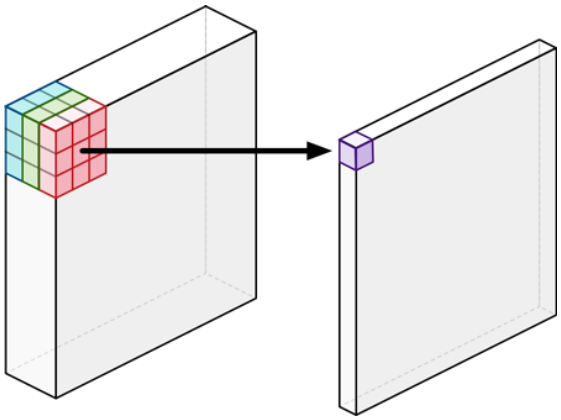

Рисунок 2.9 – Розділювальна згортка

2.2.4 Архітектура ResNet

Мережа ResNet (Residual Network або залишкова мережа) стала революційною та у 2015 році перемогла у змаганні ImageNet, обійшовши навіть архітектуру Inception V3. Автори ResNet також намагалися обійти проблему вимивання градієнту за умови збільшення кількості шарів. Оскільки згадані вище допоміжні класифікатори знизили негативний вплив глибокої мережі, проте не вирішили дану проблему остаточно, було введено поняття «залишковий блок» («residual block») .

Вважається, що накладені шари не повинні погіршувати продуктивність мережі, оскільки якщо накладати зіставлення ідектифакторів (шар, що не виконує жодних операцій), то отримана архітектура буде працювати так само. Це явище вказує на те, що більш глибока модель не повинна призводиди до більшої помилки навчання, ніж менш глибокі мережі. Автори припускають, що надати можливість стековим шарам (stacking layers) відповідати залишковому відображенню є більш простим рішенням, ніж у випадку, коли стекові шари відповідають бажаному відображенню, що лежить нижче. Залишковий блок, що розташований вище, дозволяє досягти такого результату.

Оскільки мережі з меншою кількістю шарів навчаються краще, ніж з великою, проте якості однієї невеликої мережі недостатньо, виникає питання як саме можливо оминати деякі шари моделі, не втрачаючи якість передбачення. Саме використання residual connections надає змогу пропускати шари під час навчання. Для розуміння поняття «залишку» введемо змінну x, що є вхідними даними шару та функцію розподілу $H(x)$. Різницею (або залишком) є:

$$F(x) = Output - Input = H(x) - x . \tag{2.7}$$

Переставляючи, отримуємо:

$$H(x) = F(x) + x . \tag{2.8}$$

Отже, залишковий блок намагається дізнатись істинного результат $H(x)$ та, як видно на рисунку 2.10, через наявність ідектифікаційного зв'язку, шари фактично намагаються дізнатись залишкове значення $F(x)$. У традиційній архітектурі шари досліджують значення істинного результату $H(x)$, а у залишкових мережах – $F(x)$ [32].

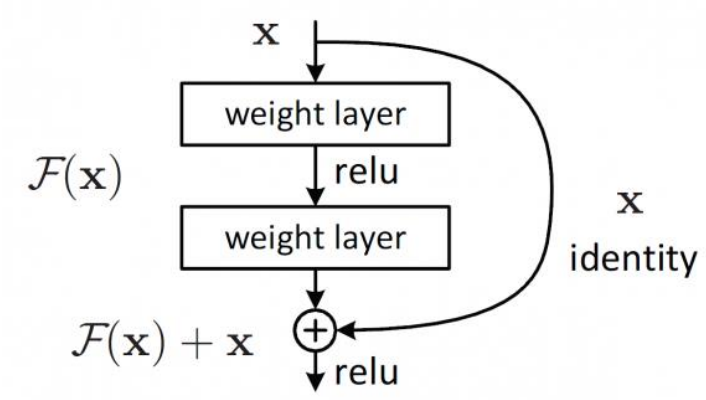

Рисунок 2.10 – Residual block

Автори покращили концепцію залишкового блоку та запропонували його попередню активацію, об'єднану з Batch Normalization. За таких умов

градієнти можуть поширюватися без перешкод через зв’язки швидкого доступу до будь-якого рівня мережі [33].

Під час тренування ResNet виконується або навчання в залишкових блоках, або пропускання навчання у таких шарах за допомогою механізму skip connections (пропускання з’єднання). Таким чином різні частини мережі тренуються з різною швидкістю в залежності від того, яким чином помилка поширюється у зворотньому напрямку. Фактично, у такому випадку мережу можна розглядати як ансабль різних моделей у наборі даних, що тренуються для отримання максимально можливої точності.

Перевагою skip connections є динамічність нейронної мережі. Оскільки заздалегідь невідома оптимальна кількість шарів (або залишкових блоків) для моделі, адже вона може залежати від складності набору даних, то за допомогою пропускання з’єднань мережа може пропускати тренування шарів, що не є корисними та не підвищують загальну точність. Таким чином, кількість шарів мережі не є важливим гіперпараметром для налаштування.

Мережі ResNet існують у різних модифікаціях та відрізняються за кількістю шарів та рівнями глибини, проте усі вони побудовані за допомогою залишкових блоків та пропусків з’єднання. Завдяки такій архітектурі, збільшення кількості шарів є легкою операцією у порівнянні з іншими типами нейронних мереж. Кожен блок ResNet має 2 (ResNet-18, 34) або 3 (ResNet-50, 101, 152) рівня глибини.

Щодо побудови фільтрів даної архітектури, правила наступні:

1. Фільтри мають розмірність 3×3.

2. Для однієї і тієї ж виходної карти об’єктів шари мають однакову кількість фільтрів.

3. Якщо розмір карти об’єктів змншується у 2 рази, кількість фільтрів збільшується у 2 рази для збереження обчислювальної складності кожного шару [34].

У даній роботі використовуються моделі на 101 та 152 шари.

2.3 Алгоритм класифікації об’єктів на зображенні

Для класифікації об’єктів на зображенні за допомогою нейронної мережі необхідно:

1. Обрати попередньо навчену модель нейронної мережі Keras Applications.
2. Виконати конфігурації моделі.
3. Завантажити тестове зображення та змінити його розміри відповідно до вимог обраної мережі.
4. Передати зображення у модель для виконання класифікації.
5. Вивести перелік рішень моделі з найбільшими ймовірністями визначених класів у форматі < «номер класу у датасеті», «ім’я класу», «значення ймовірності» >.

2.4 Метод LIME для інтерпретації рішень класифікаторів

Метод LIME у своїй назві поєднує 3 основні характеристики [35]:

1. Model-agnosticism (модельно-агностичний). Інакше кажучи, метод не залежить від моделі, передбачення якої досліджується, та не робить щодо неї прирущень. LIME розглядає її як «чорний ящик», тому єдиний спосіб зрозуміти поведінку моделі – змінювати вхідні дані та слідкувати за зміною передбачень.

2. Interpretability (інтерпретованість). Першочерговою задачею методу є зрозумілість пояснення для користувачів, внаслідок чого можуть виникати розбіжності з оригінальним простором характеристик об’єкта. Модель може мати більше сотні вхідних змінних або використовувати складні параметри, що не можуть бути інтерпретовані. Тому у поясненнях LIME використовується представлення даних, що називають інтепретованим поданням,яке відрізняється від початкового простору ознак.

3. Locality (локальність). LIME дає пояснення, виконуючи апроксимацію моделі «чорного ящика», використовуючи модель, що інтерпретується (наприклад, лінійну модель з декількома ненульовими коефіцієнтами) в околі екземпляру, який необхідно пояснити.

Таким чином, LIME генерує пояснення для рішення мережі, базуючись на компонентах моделі, що інтерпретується (наприклад, коефіцієнти лінійної регресії), яка походить на «чорний ящик» та навчається на новому поданні даних для забезпечення інтепретації.

2.4.1 Подання, що інтерпретується

Для того, щоб пояснення результату було гарантовано інтепретоване, LIME відрізняє подання, що інтерпретується, від вихідного простору ознак, що використовує модель. Подання повинно бути зрозумілим , тому його розмір не обов'язково співпадає з розміром вихідного простору ознак.

Нехай p є розмірність вихідного простіру ознак X та p' є розмірністю простору X', що інтепретується. Дані, що інтепретуються, відображаються на вхідні дані за допомогою функції відображення $h^y : X' \to X, y \in X$, що є специфічною для екземпляру, який пояснюється.

Для різних вхідних просторів використовуються разні типи відображень:

– для текстових даних можливою інтепретацією є двійковий вектор, що вказує на наявність або відсутність слова, але класифікатор може використовувати більш складні функції такі як вкладеність слів. Тоді $X' = \{0,1\}^{p'} \equiv \{0,1\} \times ... \times \{0,1\}$, де p' – кількість слів, що складають пояснюваний екземпляр, та функція відображає вектор одиниць або нулів (що вказують на наявність або відсутність слова, відповідно) у подання, що використовується моделлю. Якщо модель використовує підрахунок слів,

відображенням буде 1 на вихідну кількість слів та 0 на 0, якщо модель використовує вкладеність слів, відображення повинно конвертувати будь-яке речення, виражене як вектор одиниць та нулів, у свою вбудовану версію;

– для зображень можливим поданням є двійковий вектор, який вказує на наявність або відсутність набору суміжних схожих пікселів (суперпікселів). Тоді $X' = \{0,1\}^{p'}$, де p' – кількість суперпікселів, що були отримані за допомогою алгоритму пересегментації зображення (зазвичай, використовується алгоритм Quick Shift). Приклад розбиття наведено на рисунку 2.11. Функція відображає 1 для того, щоб залишити суперпіксель, як у вхідному зображенні, та 0 – для того, щоб залишити суперпіксель сірим (ознака відсутності);

– для табличних даних (наприклад, матриць) інтепретація залежить від типу характерних ознак: категоріальних, числових або змішаних. Для категоріальних даних $X' = \{0,1\}^{p}$, де p – кількість ознак, що використовує модель (у цьому випадку $p = p'$) та відображенням є 1 у вихідний клас екземпляру та 0 в інший клас, обраний відповідно до розподілу даних для навчання. Для числових даних $X = X'$ та відображення є ідентичним. Однак, є можливість дискретизувати числові ознаки таким чином, щоб їх можна було розглядати як категоріальні. У змішаному випадку подання є p-мірним простором, що складається з двійкових ознак, відповідних до категоріальних, та числових ознак, якщо вони не були попередньо дискретизовані.

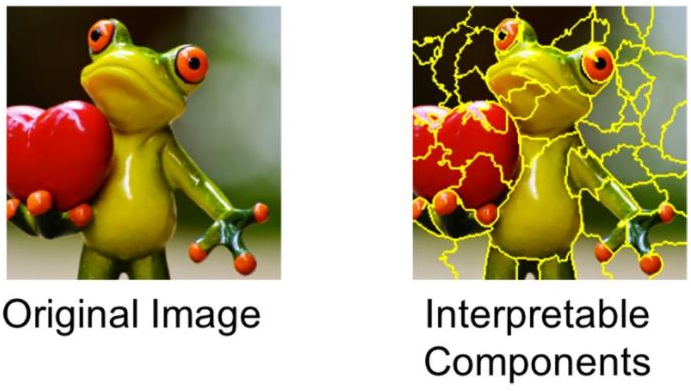

Рисунок 2.11 – Розбиття зображення на суперпікселі

Проста модель не може апроксимувати складну модель «чорного ящика», проте апроксимація її у околі окремого примірника, який необхідно інтерпретувати, є можливою. Отже, LIME припускає, що будь-яка складна модель може бути лінійною у локальному масштабі. Необхідна вагова функція $w^y : X' \to \mathbb{R}^+$, яка надає найбільшу вагу екземплярам $z \in X$, найближчим до екземпляру $y \in X$, який необхідно інтерпретувати, та найменшу вагу для екземплярів, що розташовані найдалі.

Інтепретацію можливо визначити як модель $g : X' \to \mathbb{R}, g \in G$, де G є класом «потенційно» можливих моделей для інтепретації, тобто моделей, що можуть бути представлені користувачу з використанням візуальних або текстових артефактів (наприклад, лінійні моделі або дерева рішень). Простір g є простіром X', що інтерпретується.

Нехай $L(f, g, w^y)$ є функцією втрат, що вимірює точність g у наближенні f з урахуванням вагів w^y. Оскільки не кожна $g \in G$ є достатньо простою для інтепретації, функція $\Omega(g)$ є регуляризатором, що вимірює складність (а не інтерпретованість) пояснення $g \in G$. Наприклад, для лінійних моделей $\Omega(g)$ може бути кількістю ненульових коефіцієнтів, а для дерев рішень – глибиною дерева.

2.4.2 Алгоритм роботи методу LIME

Нехай $X = \mathbb{R}^p$ є простором ознак та $y \in \mathbb{R}^p$ є вихідним поданням примірника, а $y' \in X'$ є його поданням, що інтепретується. Також визначимо $f : X = \mathbb{R}^p \to \mathbb{R}$ як модель, що пояснюється. Для задачі класифікації $f(x)$ є ймовірністю приналежності x до певного класу. Для кількох класів LIME пояснює кожен клас окремо, таким чином $f(x)$ – це передбачення певного класу. Для задачі регресії $f(x)$ є функцією регресії.

Визначимо функцію $g: X' \to \mathbb{R}$ як пояснювальну модель. Нехай $L(f, g, w^y)$ є функцією втрат, що вимірює наскільки є помилковою g у наближенні до f у локальній області, що визначена через w^y. Функція $\Omega(g)$ є мірою складності пояснення $g \in G$.

Для забезпечення інтепретованості та точності на локальному рівні необхідно мінімізувати функцію $L(f, g, w^y)$, при цьому значення $\Omega(g)$ повинно бути достатньо малим для інтепретації користувачем. Пояснення $\xi(y)$, що вироблено алгоритмом LIME, отримується таким чином:

$$\xi(y) = \arg\min_{g \in G} \left\{ L(f, g, w^y) + \Omega(g) \right\}. \quad (2.9)$$

На практиці, загальний підхід LIME до інтерпретації наступний:

1. Згенерувати N «збурених» зразків інтерпретованої версії примірника для пояснення y'. Нехай $\{z_i' \in X' \mid i = 1, ..., N\}$ є набором таких спостережень.

2. Відновити «збурених» спостережень у вихідному просторі ознак за допомогою функції відображення. Нехай $\{z_i \equiv h^y(z_i') \in X \mid i = 1, ..., N\}$ є множиною у вихідному поданні.

3. Нехай модель мережі передбачає результат кожного «збуреного» спостереження, тоді $\{f(z_i) \in \mathbb{R} \mid i = 1, ..., N\}$ є множиною відповідей.

4. Визначити вагу кожного «збуреного» спостереження, тоді $\{w^y(z_i) \in \mathbb{R}^+ \mid i = 1, ..., N\}$ є множиною ваг.

5. Розв’язати наведене вище рівняння, використовуючи датасет «збурених» зразків з їх відповідями $\{(z_i', f(z_i)) \in X' \times \mathbb{R} \mid i = 1, ..., N\}$ як дані для навчання.

Зауважимо, що складність не залежить від розміру датасету для навчання, оскільки він дає інтерпретацію для окремого рішення мережі,

натомість існує залежність від часу для обчислення прогнозів $f(z)$ та кількості зразків N.

Процес створення зразків, описаний у кроці 1, залежить від простору X', тому він відрізняється від типу вхідних даних, як вже було описано у попередньому підрозділі. Дле текстових даних або зображень, простір, що інтепретується, складається з двійкових ознак $X' = \{0,1\}^{p'}$, зразки $z_i' \in X'$ отримуються шляхом «малювання» ненульових елементів y' рівномірно та випадково, де кількість таких зразків також відбирається рівномірно. Для табличних даних крок 1 залежить від типу функцій та необхідний вхідний навчальний набір.

Оскільки для зображень простір, що інтерпретується, є двійковим вектором, що вказує на наявність або відсутність суперпікселя, процес полягає у випадковому «зафарбовуванні» деяких суперпікселів у сірий колір. Іншими словами, якщо необхідно пояснити рішення мережі для зображення, проводять «збурення» зображення та отримують прогнози на зразках, що мають один або кілька прихованих суперпікселів (рис. 2.12).

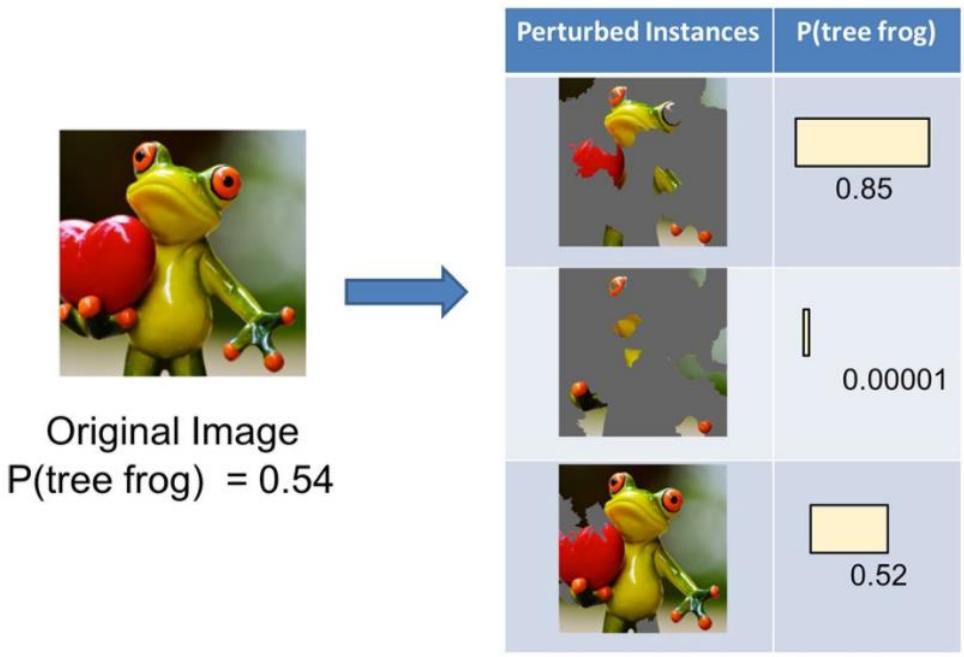

Рисунок 2.12 – Приклади «збурених» екземплярів зображення

2.4.3 Розріджені лінійні інтерпретації

Марко Рібейро та ін. сфокусувались на понятті розріджених лінійних інтепретацій, що відповідають конкретним виборам сімейства пояснень G, функції втрат L та регуляризації Ω [36]. Явно цей вибір не обґрунтовано, але можна зробити висновок про деякі ймовірні причини цього вибору:

1. Клас лінійних моделей як сімейство пояснень G $g(z') = \beta \cdot (z')$. Лінійні моделі дозволяють виміряти вплив кожної ознаки на рішення моделі, перевіряючи як власне значення, так і його знак. Таким чином, лінійні моделі забезпечують якісне розуміння між змінними пояснення та відповідью, що робить їх потенційно інтепретованими моделями, що підходять для використання у якості моделей пояснення.

2. Експоненційне ядро, визначено на деякій функції відстані D як вагова функція $w^y(z) = e^{\{\frac{-D(y,z)^2}{\sigma^2}\}}$.

Функції відстані, що використовуються є косинусною мірою сходження для текстових даних та евклідовою відстанню для зображень та табличних даних. Проте у реалізації LIME на Python та R користувач може використати будь-яку іншу відстань.

3. Зважена помилка найменших квадратів як функція втрат:

$$L(f, g, w^y) = \sum_{i=1}^{N} w^y(z_i)(f(z_i) - g(z_i'))^2, \qquad (2.10)$$

де $(z_i', f(z_i))$ є набором даних «збурених» екземплярів з їх відповідями, визначинеми вище. Квадратичні функції частіше більш піддаються математичній обробці, ніж інші функції втрат.

3. Обмеження до K число ненульових коефіцієнтів як регуляризатору.

$$\Omega(g) = \infty 1_{\{\|\beta\|_0 > K\}} \equiv \begin{cases} \infty, \text{if } \|\beta\|_0 > K\} \\ \infty, \text{if } \|\beta\|_0 \leq K\} \end{cases}, \quad (2.11)$$

де $\|\beta\|_0 = \sum_j |\beta|^0$ є $L0$-«нормою», тобто кількістю ненульових елементів.

Параметр регуляризації K має вирішальне значення для задачі пояснення, оскільки він встановлює кількість суперпікселей, що будуть представлені користувачу.

Існує два важливих гіперпараметри: ширина ядра σ та кількість суперпікселів, що будуть використовуватися для пояснення. Отже, рівняння, що необхідно вирішити для отримання інтерпретації за допомогою LIME, виглядає як:

$$\xi(y) = \arg\min\{\sum_{i=1}^{N} e^{\frac{-D(y,z)^2}{\sigma^2}} (f(z_i) - \beta \cdot z_i')^2 + \infty 1_{\{\|\beta\|_0 > K\}}\}. \quad (2.12)$$

Рівняння неможливо вирішити безпосередньо, тому автори спочатку апроксимують рішення, для початку обравши K суперпікселей, а потім оцінюючи коефіцієнти за допомогою зважених найменших квадратів.

Таким чином, підхід, що використовується зараз у методу LIME для апроксимації рішення попереднього рівняння та отримання інтепретації наступний:

1. Згенерувати N «збурених» зразків інтерпретованої версії примірника для пояснення y'. Нехай $\{z_i' \in X' \mid i = 1,..., N\}$ є набором таких спостережень.

2. Відновити «збурених» спостережень у вихідному просторі ознак за допомогою функції відображення. Нехай $\{z_i \equiv h^y(z_i') \in X \mid i = 1,..., N\}$ є множиною у вихідному поданні.

3. Нехай модель мережі передбачає результат кожного «збуреного» спостереження, тоді нехай $\{f(z_i) \in \mathbb{R} \mid i = 1,..., N\}$ є множиною відповідей та

нехай $\gamma = \{(z_i', f(z_i)) \in X' \times \mathbb{R} \mid i = 1,...,N\}$ є множиною «збурених» зразків з відповідями.

4. Визначити вагу кожного «збуреного» спостереження, тоді $\{w^y(z_i) \in \mathbb{R}^+ \mid i = 1,...,N\}$ є множиною ваг.

5. Визначити K суперпікселів, що найкращим чином інтерпретують результат моделі «чорного ящика» з набору даних γ.

6. Передати у зважену модель лінійної регресії датасет, що складається з K ознак, обраних на кроці 5. Якщо модель «чорного ящика» є регресією, лінійна модель безпосередньо прогнозує результати моделі, що інтерпретується. Якщо модель є класифікатором, то лінійно модель прогнозує ймовірність обраного класу.

7. Вилучити коефіцієнти лінійної моделі та використовувати їх у якості пояснення локальної поведінки досліджуваної моделі.

На рисунку 2.13 зображено приклад даної процедури, для якої попередньо визначено кількість суперпікселів для створення інтерпретації.

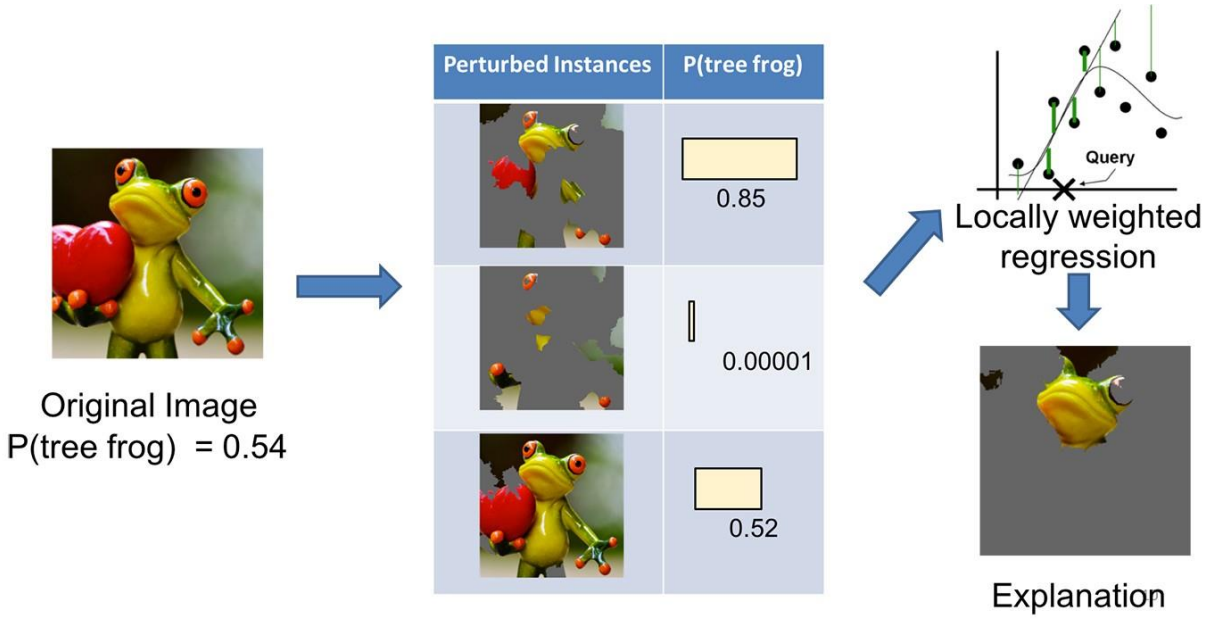

Рисунок 2.13 – Пояснення рішення класифікатора за допомогою LIME

2.5 Алгоритм інтерпретації рішень моделі методом LIME

Для проведення інтепретації рішень моделі за допомогою методу LIME необхідно:

1. Завантажити бібліотеку lime для проведення подальшої інтепретації.
2. Передати до алгоритму масив рішень, що були спрогнозовані, та обрати бажану кількість рішень з найбільшою ймовірністю для подальшого дослідження.
3. Передати до алгоритму класифікатор, що виконував прогнозування рішення.
4. Вказати розмір околу для тренування лінійної моделі.
5. Обрати з масиву рішення, яке необхідно пояснити.
6. Вказати кількість характерних ознак (суперпікселів) для виведення результату інтепретації.
7. Вказати, чи необхідно виводити ті ознаки, що негативно вплинули на рішення моделі.
8. Вказати формат виведення результату інтерпретації: значущі ознаки та прихована решта зображення, що не відноситься до пояснення, чи усе зображення з виділеними ознаками.
9. Вивести результат інтерпретації рішення моделі.

Таким чином, у розділі було розглянуто модель CNN, бібліотеку Keras та аспекти роботи з нею, різновиди згорткових нейронних мереж, передумови їх створення, характерні особливості будови. Також було вивчено алгоритм LIME для інтерпретації рішень нейронних мереж, поняття вихідного подання та подання, що інтепретується, розріджені лінійні інтепретації.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОСЛІДЖЕНЬ МОДЕЛЕЙ КЛАСИФІКАЦІЇ

3.1 Реалізація алгоритмів класифікації об’єктів та інтерпретації моделей мовою Python

3.1.1 Налаштування середовища

В першу чергу слід обрати та налаштувати середовище, у якому буде проведена подальша реалізація. Для даної роботи було обрано хмарне середовище Google Colaboratory. Для початку роботи необхідно:

1. Створити новий файл Colaboratory з розширенням .ipynb.

2. Налаштувати середовище виконання (runtime type), обравши у меню RunTime –> Change runtime type. У вікні налаштувань обрати виконання на графічному процесорі GPU та зберегти налаштування.

Далі для налаштування проводиться завантаження необхідних пакетів та бібліотек:

1. Lime – пакет Python API для роботи з алгоритмом LIME.

2. Matplotlib – бібліотека для візуалізації зображення.

3. Tensorflow – бібліотека для імпорту Keras Applications (моделей нейронних мереж).

4. NumPy – бібліотека для проведення обчислень.

3.1.2 Класифікація зображення

Для проведення класифікації необхідно запрограмувати наступні кроки:

1. Завантажити зображення у середовище за допомогою модуля wget, що зберігає файл, який розміщується за посиланням. Достатньо встановити модуль та передати посилання, що починається із «https://…» та бажане зображення буде збережено із оригінальною назвою.

2. Завантажити модель нейронної мережі з використанням псевдонімів (aliases). В залежності від бажаної моделі буде змінюватись остання частина псевдоніму, а «tf.keras.applications» – залишиться незмінною. Нижче на рисунку 3.1 наведено фрагмент коду для завантаження мережі ResNet-152.

```
model = tf.keras.applications.resnet_v2.ResNet152V2 (include_top=True, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)
```

Рисунок 3.1 – Фрагмент коду завантаження моделі ResNet-152

Також встановлюється значення аргументів:

– include_top – встановлюється у значення True, якщо необхідно включати повнозв’язний шар в кінці мережі;

– weights – обирається параметр None, якщо ініціалізація вагів повинна бути виконана випадково (рандомно), 'imagenet', якщо необхідно використовувати ваги після попереднього навчання мережі на датасеті ImageNet, або можливо додати адресу до файлу з вагами, які необхідно завантажити до моделі;

– input_tensor – необов’язковий тензор Keras для введення зображення для моделі;

– input_shape – опціональне встановлення розмірів зображення, повинно бути визначено лише при значенні include_top=False, у іншому випадку на вхід необхідно подати екземпляр у форматі (224, 224, 3), де 224 – ширина та висота зображення, а 3 – кількість вхідних каналів. Формат може бути представлений у зворотній формі;

– pooling – параметр використовується, якщо значення include_top=False, встановлюється None, якщо виходом моделі є чотирьохмірний тензор останнього шару згортки, avg – якщо виходом буде двомірний тензор останнього шару згортки, max – якщо необхідно виконасти максимальний пулінг;

– classes – опціональне значення кількості класів, вказується лише у випадку, коли include_top=True та не вказано значення аргумента weights.

3. Зконфігурувати модель за допомогою функції compile, що приймає аргументи (рис. 3.2):

– optimizer – ім’я оптимізатору для мережі, у даній роботі для усіх мереж використовується оптимізатор Adam;

– loss – ім’я функції втрат, у роботі використано функцію категоріальної крос-ентропії (перехресної ентропії);

– metrics – перелік метрик, які оцінюються моделлю під час навчання та тестування. Зазвичай, використовуються метрики accuracy.

Інші параметри є опціональними.

```
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
```

Рисунок 3.2 – Фрагмент коду конфігурації моделі ResNet-152

4. Завантажити зображення та змінити його розміри до параметрів, необхідних для обраної мережі (рис. 3.3):

```
image = plt.imread("chihuahua-portrait.jpg")
from skimage import transform
def preprocess(image):
image = transform.resize(image, (224, 224, 3))
image = np.expand_dims(image, axis=0)
return image
images = preprocess(image)
```

Рисунок 3.3 – Фрагмент коду завантаження та зміни зображення

Зміна параметрів зображення виконується за допомогою функції resize, що містить параметри висоти, ширини та кількості каналів. Для мереж ResNet та MobileNet розмір зображень повинен становити 224×224 пікселів, для мережі Inception – 299×299 пікселів.

5. Передати змінене зображення до мережі, виконати класифікацію за допомогою функції predict та, використовуючи функцію decode_predictions, вивести перелік рішень моделі з найбільшими ймовірністями визначених

класів у форматі < «номер класу у датасеті», «ім’я класу», «значення ймовірності» > (рис. 3.4).

```
pred = model.predict(images)
from keras.applications.inception_v3 import decode_predictions
for x in decode_predictions(pred, top=3)[0]:
print(x)
```

Рисунок 3.4 – Фрагмент коду обробки зображення мережею

3.1.3 Інтерпретація рішень моделі

Для застосування алгоритму LIME до результату класифікації, необхідно:

1. Створити екземпляр класу LimeImageExplainer(), що пояснює рішення моделей (рис. 3.5).

```
from lime import lime_image
explainer = lime_image.LimeImageExplainer()
```

Рисунок 3.5 – Фрагмент коду створення екземпляру класу

2. Викликати метод класу LimeImageExplainer() explain_instance для генерації пояснень рішення (рис. 3.6). Аргументи методу:

– image – зображення, що необхідно інтерпретувати;

– classifier_fn – функція ймовірності рішення класифікатора, що приймає масив Numpy та виводить ймовірності рішень моделі;

– top_labels – кількість рішень мережі з найбільшими значеннями ймовірності, що необхідно пояснити;

– hide_color – визначає колір забарвлення областей зображення, що не відносяться до пояснення;

– num_samples – розмір околу для навчання лінійної моделі.

```
explanation = explainer.explain_instance(images[0],
model.predict, top_labels=3, hide_color=0, num_samples=1000)
```

Рисунок 3.6 – Фрагмент коду генерації пояснень рішення

3. Застосувати метод get_image_and_mask для задання налаштування інтепретації та вивести результат (рис. 3.7). Функція містить такі параметри:

– label – рішення мережі, що необхідно пояснити;

– positive_only – приймає значення True або False, якщо встановити True, то будуть відображені лише ті суперпікселі, що сприяють рішенню мережі;

– hide_rest – приймає значення True або False, якщо встановити True, то частина зображення, що не відноситься до інтепретації, буде зафарбована;

– num_features – бажана кількість суперпікселів, що необхідно включити до інтепретації.

```
temp, mask = explanation.get_image_and_mask
(explanation.top_labels[0], positive_only=True, num_features=5,
hide_rest=True)
plt.imshow(mark_boundaries(temp, mask))
```

Рисунок 3.7 – Фрагмент коду налаштування інтепретації

Функція mark_boundaries повертає зображення з виділенними межами фрагментів.

3.2 Ілюстрація інтерпретації моделей методом LIME

У даному підрозділі наведено результати практичного застосування алгоритму LIME для інтепретації моделей на результатах класифікації зображень за допомогою чотирьох нейронних мереж Keras Applications: MobileNet V1, Inception V3, ResNet-101 V2 та ResNet-152 V2. Параметри використаних моделей наведено у таблиці 3.1 [37].

Таблиця 3.1 – Загальна середня оцінка роботи моделей

Назва моделі	Розмір, Мб	Найвища точність	Кількість параметрів	Глибина
Inception V3	92 Мб	0,937	23,851,784	159
MobileNet V1	16 Мб	0,895	4,253,864	88
ResNet-101 V2	171 Мб	0,938	60,380,648	—
ResNet-152 V2	232 Мб	0,942	23,851,784	159

На рисунку 3.8 наведено тестове зображення, що буде завантажено до кожної мережі.

Рисунок 3.8 – Тестове зображення

На рисунку 3.9 перелічені перші три найбільш ймовірні класифікації, зроблені мережею Inception V3. Класифікація правильна, мережа «впевнена» на 87%, що на зображенні знаходиться об’єкт «Чіхуахуа».

```
('n02085620', 'Chihuahua', 0.8775373)
('n02087046', 'toy_terrier', 0.008931621)
('n02113978', 'Mexican_hairless', 0.003832607)
```

Рисунок 3.9 – Топ-3 рішень моделі Inception V3

На рисунку 3.10 представлені результати роботи алгоритму LIME з налаштуванням на 5 та 7 суперпікселів, що складають пояснення рішення моделі. Як видно з інтепретації, Inception звертає увагу на характерні області об’єкту, що дійсно характерні для даної класифікації та із зростанням кількості суперпікселів пояснення не втрачає зрозумілості, модель не включає зайві фрагменти.

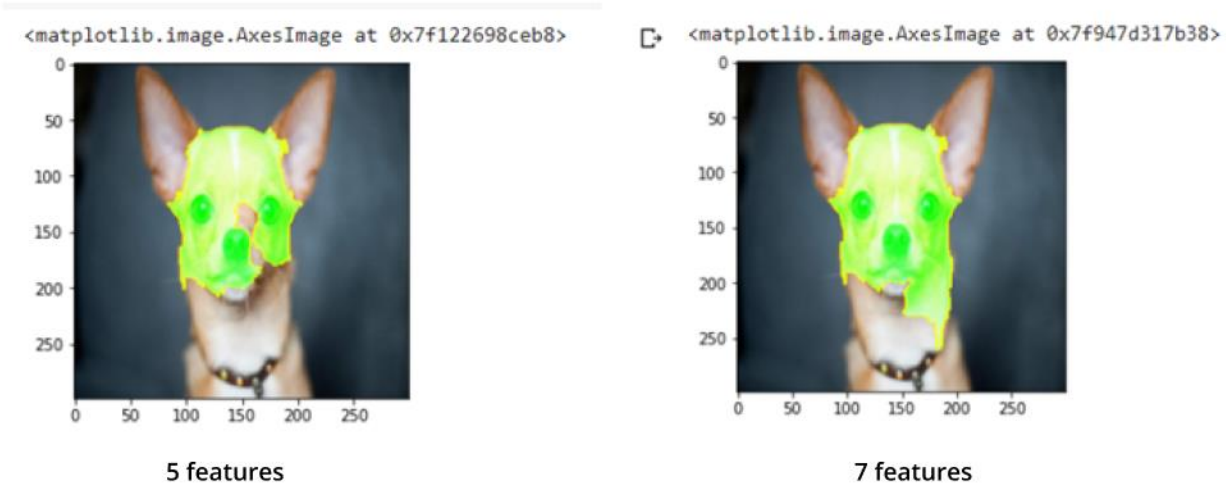

Рисунок 3.10 – Інтерпретація рішення моделі Inception V3

На рисунках 3.11 та 3.12 представлені результати класифікації мережею MobileNet V1 та пояснення рішень з тією ж кількістю значущих фрагментів. Мережа показала вищу ймовірність (97%), але не «звернула увагу» на око об’єкта у поясненні з 5 суперпікселів та виділила, навпаки, зайві фонові області, що не відносяться до собаки. Така інтепретація викликає сумніви у якості мережі.

```
('n02085620', 'Chihuahua', 0.9756594)
('n02113023', 'Pembroke', 0.016805632)
('n02087046', 'toy_terrier', 0.005420986)
```

Рисунок 3.11 – Топ-3 рішень моделі MobileNet V1

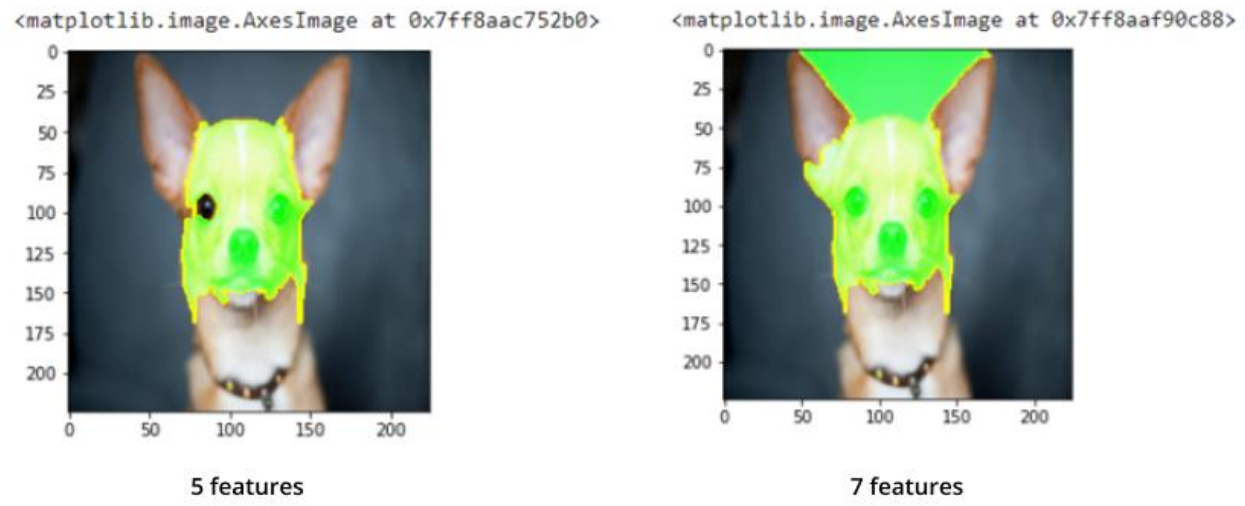

Рисунок 3.12 – Інтерпретація рішення моделі MobileNet V1

Залишкові мережі ResNet-101 та ResNet-152 показали максимальну ймовірність при класифікації у 99% (рис. 3.13).

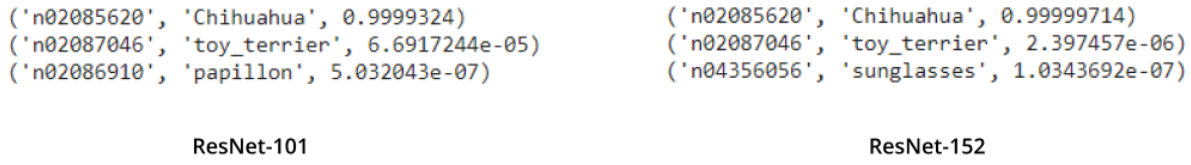

Рисунок 3.13 – Топ-3 рішень моделей ResNet-101 V2 та ResNet-152 V2

Модель на 101 шар показала нижчу якість інтерпретації, ніж на 152 шари через включення фону у склад значущих суперпікселів (рис. 3.14).

При створенні інтепретації для мережі ResNet-152 навіть після збільшення кількості суперпікселів виділеними залишились лише фрагменти класифікованого об’єкта (рис. 3.15). У Додатку А. наведено приклади інтепретації рішень моделі ResNet-101 V2 методом LIME з використанням зображень підвищенної складності.

Слід зазначити, що така поведінка моделей є лише одним з прикладів даного дослідження та не є доказом гарантованої вдалої або суперечливої інтепретації, що буде наведено далі у експертному оцінюванні.

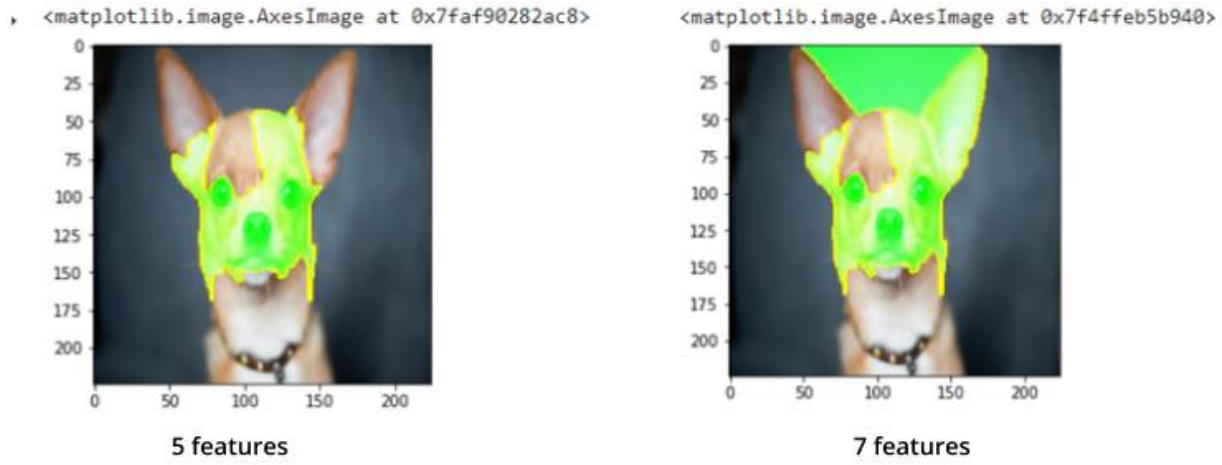

Рисунок 3.14 – Інтерпретація рішення моделі ResNet-101 V2

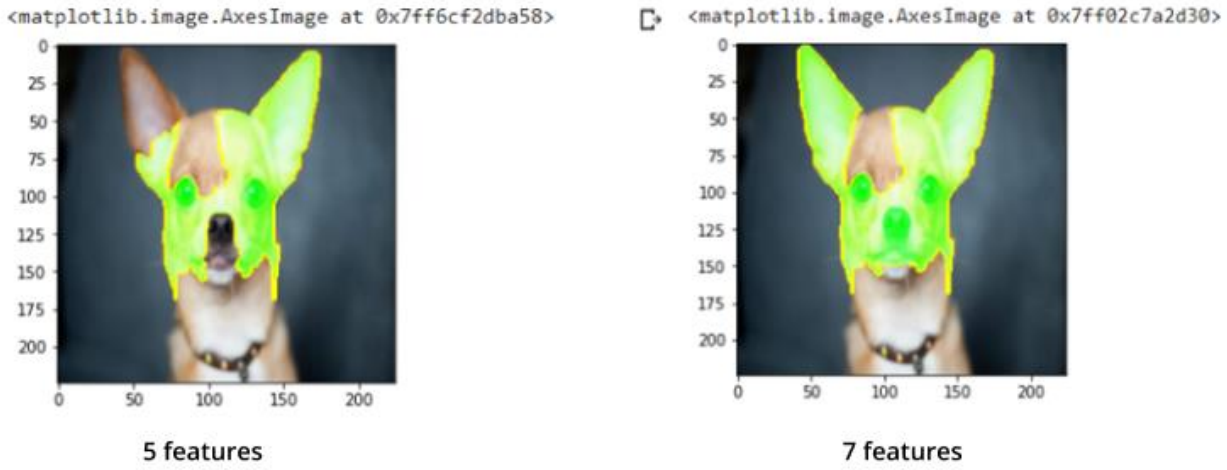

Рисунок 3.15 – Інтерпретація рішення моделі ResNet-152 V2

3.3 Аналіз якості класифікації за допомогою метрик *precision*, *recall* та *F* -міри

Для оцінки отриманих результатів моделей було проведено розрахунки метрик *precision* (точність), *recall* (повнота) та *F* -міри.

Усі три показники залежать від чотирьох параметрів: True Positive (TP) (істинно позитивний), True Negative (TN) (істинно негативний), False Positive (FP) (помилково позитивний), False Negative (FN) (помилково негативний). Параметр True Positive вказує на кількість випадків, коли об’єкт

дійсно існує на зображенні та модель його вірно класифікує. False Positive вказує на те, що об’єкт присутній на зображенні, але був класифікований невірно. Параметр False Negative вказує на кількість об’єктів, що повинні були бути класиіфіковані, але модель їх не знайшла на зображенні. Параметр True Negative не відповідає даному дослідженню та не був розрахований.

Метрика *precision* може бути інтерпретована як частка об’єктів, що були правильно класифіковані та дійсно існують на зображенні датасету. Метрика розраховується за формулою:

$$precision = \frac{TP}{TP + FP}. \tag{3.1}$$

Метрика *recall* показує частку об’єктів, яку класифікував алгоритм з усієї кількості обєктів класу

$$recall = \frac{TP}{TP + FN}. \tag{3.2}$$

F-міра є об’єднанням *precision* та *recall* та характерезує якість моделі в залежності від обох показників. У випадку, коли F-міра досягає значення 1, кажуть, що модель якісна. У випадку, коли вона досягає значення 0, кажуть, що один з аргументів (або точніть, або повнота) є низьким і модель вважається неякісною. Для розрахунку F-міри використовується показник β, що є середнім гармонічним [38]. Зазвичай $\beta = 1$. Формула розрахунку F-міри:

$$F_\beta = (1 + \beta^2)\frac{precision \times recall}{(\beta^2 \times precision) + recall}. \tag{3.3}$$

У даному дослідженні результат класифікації з ймовірністю менше 50% не вважається позитивним та відноситься до категорії випадків False Negative.

У таблиці 3.2 наведено загальну середню оцінку кожної моделі.

Таблиця 3.2 – Загальна середня оцінка роботи моделей

Назва моделі	*precision*, %	*recall*, %	F – міра
Inception V3	0,94	0,67	0,78
MobileNet V1	0,94	0,63	0,75
ResNet-101 V2	0,75	0,9	0,82
ResNet-152 V2	0,78	0,82	0,8

Отже, за результатами оцінки моделей за метрикою *recall* найбільшу кількість об'єктів були класифіковано залишковими мережами, а мережі Inception та MobileNet частіше не знаходили об'єкт на зображенні взагалі (вплив великої кількості False Negative результатів). За метрикою *precision*, навпаки, першість за кількістю правильно класифікованих об'єктів у мереж Inception та MobileNet, даний показник є нижчим для мереж ResNet через велику кількість False Positive результатів. Отже, за відношенням *precision* та *recall* (F-мірою) найбільш якісними є мережі ResNet.

На результати значно вплинула вибірка зображень, що складалась не тільки з простих зображень з об'єктом на однотонному фоні. Частина тестових примірників була взята з датасетів зображень для змагальних атак та містила нечіткі об'єкти, схожі на інші предмети.

3.4 Аналіз якості інтерпретації моделей за допомогою експертних оцінок

Для оцінки інтепретованості рішень моделей було проведено опитування з метою виявлення нейронної мережі, результати інтепретації якої є найбільш зрозумілими та пояснюваними для користувачів. Кожне питання складається з рішення моделі та тестового зображення, що є результатом

роботи алгоритму LIME. На зображенні виділено 5 найбільш значущих фрагментів, решта площі зображення зафарбована чорним кольром.

Експертам було запропоновано оцінити, чи можуть вони на підставі власного досвіду визначити на зображенні:

1. Індивідуальні характерні ознаки об’єкта (характерні деталі, наприклад, ніс, рот, очі, їх розміри та взаємне розташування).

2. Текстури.

3. Характерні кольори.

4. Загальні характерні ознаки об’єкта (загальні пропорції, світло-тіньові обриси).

Кожне зображення необхідно було оцінити від 0 до 4 балів:

– 4 – якщо за фрагментами можливо визначити усі ознаки об’єкта (пункти 1-4);

– 3 – якщо за фрагментами можливо визначити лише ознаки об’єкта, визначені пунктами 1-3;

– 2 – якщо за фрагментами можливо визначити лише ознаки об’єкта, визначені пунктами 1-2;

– 1 – якщо за фрагментами можливо визначити лише ознаки об’єкта, визначені пунктом 1;

– 0 – виділені фрагменти повністю не відповідають запропонованій назві об'єкта.

Експерти оцінювали 8 різних зображень, що були класифіковані чотирма нейронними мережами, всього опитування налічує 32 питання. Усього в опитуванні прийняли участь 61 експерт. Завдання анкети та приклад запитання наведено у додатку Б. Надалі для досліджуваних мереж застосована наступна нумерація:

1. MobileNet V1.

2. Inception V3.

3. ResNet-101.

4. ResNet-152.

В результаті узагальнення всіх даних опитування експертів щодо відповідності визначених значущих фрагментів зображення наданій нейронними мережами класифікаціїї отримаємо таблицю 3.3,

де C_{ij} – ранг, що виставлений i-тим експертом j-му зображенню;

$x_1, x_2, ...x_j, ...x_n$ – код зображення;

n – кількість зображень;.

m – кількість експертів.

Таблиця 3.3 – Зведена відомість про рейтингові оцінки, що виставлені групою експертів при оцінці фрагментів зображень

Експерт	Оцінка, надана експертами					
	x_1	x_2	…	x_j	…	x_n
1-й	C_{11}	C_{12}	…	C_{1j}	…	C_{1n}
2-й	C_{21}	C_{22}	…	C_{2j}	…	C_{2n}
…	…	…	…	…	…	…
i	C_{i1}	C_{i2}	…	C_{ij}	…	C_{in}
…	…	…	…	…	…	…
m	C_{m1}	C_{m2}	…	C_{mj}	…	C_{mn}

Після отримання цих даних необхідно визначити узагальнений ранг. Зазавичай для цього використовують метод середнього аріфметичного рангу [39]. Для визначення середнього значення j-того зображення використовується формула:

$$\bar{C}_j = \frac{1}{m}\sum_{i=1}^{m} C_{ij}. \qquad (3.4)$$

Слід зазначити, що сучасні дослідження теорії вимірів та експертних оцінок [40] звертають увагу на те, що метод середнього аріфметичного не повною мірою відбиває думку експертів. Наприклад, зображенню лисиці звичайної, що була класифікована мережею ResNet-152, експерти проставили

наступні бали: 32 експерти вважають, що на зображені наявні все необхідні характеристики (4 бали), 24 – вважають, що загальні риси об’єкта неможливо визначити (3 бали), 3 експерти знайшли лише індивідуальні ознаки та текстури (2 бали) та 2 останніх опитаних проставили 0 балів, не знайшовши жодної характерної риси на фрагментах. Метод середніх дає загальний ранг 3,38, що не відповідає дійсності.

Найбільш інформативний в цьому випадку є метод медіан. При визначенні медіани оцінки висталяються у порядку зростання, після чого у випадку непарної кількості оцінок береться та, що за порядком є центральною. Якщо кількість оцінок парна, то береться два значення з першої половини (ліва медіана), або з другої половини оцінок (права медіана). Використання методу медіани у наведеному прикладі дає середнє значення 4, що є вірним.

В результаті проведення опитування були отримані вихідні дані експертизи, що наведені у таблиці 3.4. Для зручності було проведенно транспонування таблиці 3.3.

Таблиця 3.4 – Первинні результати проведення експертизи

Результат класифікації	Номер мережі	Оцінки, надані експертами								
		1	2	3	4	5	…	45	…	61
Лисиця звичайна 1	1	0	0	0	0	0	…	0	…	0
Лисиця звичайна 1	2	0	0	0	1	0	…	0	…	0
Лисиця звичайна 1	3	0	0	0	0	0	…	0	…	0
Лисиця звичайна 1	4	0	0	0	0	0	…	0	…	0
Бельгійська вівчарка	1	3	2	3	4	3	…	3	…	3
…	…	…	…	…	…	…	…	…	…	…
Парасоля 2	2	2	2	4	4	1	…	3	…	0
Чіхуахуа	3	2	3	3	4	1	…	3	…	3
Чіхуахуа	4	1	1	2	4	1	…	3	…	3

Результати підрахунку однакових рангів наведено у таблиці 3.5.

Таблиця 3.5 – Результати підрахунку однакових рангів

Результат класифікації	Номер мережі	Оцінки експертів				
		0	1	2	3	4
Лисиця звичайна 1	1	51	6	4	0	0
Лисиця звичайна 1	2	50	7	4	0	0
Лисиця звичайна 1	3	49	10	2	0	0
Лисиця звичайна 1	4	30	14	13	3	1
Бельгійська вівчарка	1	0	1	3	17	40
…	…	…	…	…	…	…
Парасоля 2	2	5	10	18	17	11
…	…	…	…	…	…	…
Чіхуахуа	3	0	1	6	22	32
Чіхуахуа	4	0	5	8	23	25

У таблиці 3.6 наведено результати визначення позиції рангового числа відповідно до первинного ранжирування експерта за середнім арифметичним рангом, у таблиці 3.7 – за методом медіан.

Таблиця 3.6 – Визначення позиції рангового числа відповідно до первинного ранжирування експерта за середнім арифметичним рангом

Результат класифікації	Номер мережі	Номер мережі			
		1	2	3	4
Лисиця звичайна 1	1	0,23	—	—	—
Лисиця звичайна 1	2	—	0,25	—	—

Кінець таблиці 3.6

Результат класифікації	Номер мережі	Номер мережі			
		1	2	3	4
Лисиця звичайна 1	3	—	—	0,23	—
Лисиця звичайна 1	4	—	—	—	0,87
Бельгійська вівчарка	1	3,57	—	—	—
…	…	…	…	…	…
Парасоля 2	2	2,31	—	—	—
…	…	…	…	…	…
Чіхуахуа	3	—	—	3,39	—
Чіхуахуа	4	—	—	—	3,11

Таблиця 3.7 – Визначення позиції рангового числа відповідно до первинного ранжирування експерта за методом медіан

Результат класифікації	Номер мережі	Номер мережі			
		1	2	3	4
Лисиця звичайна 1	1	0	—	—	—
Лисиця звичайна 1	2	—	0	—	—
Лисиця звичайна 1	3	—	—	0	—
Лисиця звичайна 1	4	—	—	—	1
Бельгійська вівчарка	1	4	—	—	—
…	…	…	…	…	…
Парасоля 2	2	2	—	—	—
Чіхуахуа	3	—	—	4	—
Чіхуахуа	4	—	—	—	3

Результати експертизи за методом середніх арифметичних рангів та методом медіан узагальнені у таблиці 3.8.

Таблиця 3.8 – Узгоджені результати експертизи

	Номер мережі			
	1	2	3	4
Сума середніх значень	19,74	21,4	19,42	18,56
Сума медіан	20	22	21	19
Ранжирування за середнім значенням	2	1	3	4
Ранжирування за медіаною	3	1	2	4

Результати аналізу даних експертизи за методом медіан і методом середніх виявились не суперечливими. За обома методами мережа 2 виявилась найбільш придатною до зрозумілої для експертів інтепретації. Також експертна оцінка визначила модель 4 як мережу з найменш зрозумілими результатами інтепретації. Не зважаючи на високу точність класифікації за метриками, за методом LIME мережа іноді обирала нехарактерні області як найбільш значущі, що значно знизило її якість для експертів. Щодо мереж 1 та 3, вони займають другу та третю позиції з дуже малою різницею, можна вважати, що вони однакові за інтепретованістю за результатами опитування. Отже, досліджувані нейронні мережі отримали ранжирування $\left[4 < \{1,3\} < 2\right]$.

Таким чином було проведено експертне оцінювання інтепретованості результатів моделей нейронних мереж Keras Applications, найбільш зрозумілими згідно опитування виявились зображення, класифіковані мережею Inception V3.

ВИСНОВКИ

У рамках представленого дослідження було розроблено застосунок для класифікації об’єктів на зображенні з використанням бібліотеки Keras та бібліотеки LIME для інтерпретацій рішень моделей. До основних розглянутих теоретичних та практичних питань можна віднести:

– сучасний стан вирішення питання класифікації об’єктів на зображенні;

– структура згорткових мереж;

– нейронні мережі Inception V3, MobileNet та ResNet, їх покоління, характерні особливості структури;

– бібліотека Keras, функціонал для створення та налаштування нейронних мереж;

– моделі Keras Applications, попередньо навчені для класифікації об’єктів на зображенні;

– мова Python, її модулі та бібліотеки, необхідні для роботи з нейронними мережами;

– бібліотека LIME, функціонал для інтерпретацій рішень моделей, що класифікують об’єкти на зображеннях;

– метрики *precision*, *recall* та F-міра для оцінки якості моделей;

– метод середнього арифметичного рангу та метод медіан для визначення узагальненного рангу згідно експертної оцінки.

У результаті практичних досліджень було оцінено якість класифікації нейронними мережами та зрозумілість інтепретації результатів. Базуючись на розрахунках точності моделей визначено, що кращими за показником *precision* є мережа Inception V3, та за оцінкою експертів найбільш зрозумілими є інтепретації методом LIME до результатів класифікації тієї ж моделі.

Для розробки застосунку було використано сучасну бібліотеку Keras та мову Python. Обидва програмні засоби мають офіційну деталізовану

документацію, засоби для візуалізації реалізованих алгоритмів, методи для створення і навчання нейронних мереж і високу швидкість обробки коду.

Результати були апробовані на VI-й Міжнародній науково-технічній конференції «Інформатика, управління та штучний інтелект» [20], на XI-й Міжнародній науково-практичній конференції «Free and Open Source Software» [25], на XXIV-ому Міжнародному молодіжному форуму [31]. На конференції у рамках XXIV-го Міжнародному молодіжному форуму робота посіла II місце.

У подальшому є можливість розвивати дане дослідження у напрямку детальної інтерпретації моделей на рівні окремих шарів для аналізу рішень на проміжних кроках. Також можливо ускладнити задачу до класифікації кількох об’єктів на зображенні та інтерпретації кожного окремого рішення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Most Exciting Uses of Image Recognition That are Already Changing Our Lives. URL: https://indatalabs.com/blog/uses-image-recognition, (дата звернення: 26.10.2020).

2. Different Computer Vision Tasks. URL: https://medium.com/@ananya.banerjee.rr/different-computer-vision-tasks-b3b49bbae891, (дата звернення: 26.10.2020).

3. Facial recognition: top 7 trends (tech, vendors, markets, use cases and latest news). URL: https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition, (дата звернення: 26.10.2020).

4. Інформаційні системи та технології в управлінні. Методичні вказівки,теоретичні відомості і завдання до лабораторних робіт для студентів та магістрів денної форми навчання спеціальності 7.803060101 Менеджмент організацій і адміністрування. Частина 3. Класифікація в бізнес-аналітиці/уклад. Біла Н.І. – Запоріжжя: ЗНТУ, 2014. – с. 50. URL: http://eir.zntu.edu.ua/bitstream/123456789/342/1/met_vk_ bila_3.pdf, (дата звернення: 26.10.2020).

5. М. Н. Краснянский, А. Д. Обухов. Сравнительный анализ методов машинного обучения для решения задачи классификации документов научно-образовательного учреждения. Вестник ВГУ, Серия: Системный анализ и информационные технологии, 2018, с.173–182. №3 URL: http://www.vestnik.vsu.ru/pdf/analiz/2018/03/2018-03-19.pdf, (дата звернення: 25.10.2020).

6. Titanic: Machine Learning from Disaster. Kaggle, 2012. URL: https://www.kaggle.com/francksylla/titanic-machine-learning-from-disaster/data, (дата звернення: 22.10.2020).

7. Готовим данные для анализа правильно. Data Mining. Машинное обучение. Habr, 2017. URL: https://habr.com/ru/post/342366/, (дата звернення: 22.10.2020).

8. Методы отбора фич. Habr, 2015. URL: https://habr.com/ru/post/264915/, (дата звернення: 23.10.2020).

9. Архитектуры нейросетей. URL: https://habr.com/ru/company/nix/blog/430524/, (дата звернення: 08.11.2020).

10. Transfer learning from pre-trained models. URL: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751, (дата звернення: 08.11.2020).

11. Объясняем предсказания вашей нейронной сети. Часть 1. URL: https://dou.ua/lenta/articles/interpreting-machine-learning-1/, (дата звернення: 26.10.2020).

12. Top 9 Machine Learning Applications in Real World. URL: https://data-flair.training/blogs/machine-learning-applications/, (дата звернення: 26.10.2020).

13. Adversarial Examples. URL: https://christophm.github.io/interpretable-ml-book/adversarial.html, (дата звернення: 26.10.2020).

14. Model-Agnostic Methods for Interpreting any Machine Learning Model. URL: https://towardsdatascience.com/model-agnostic-methods-for-interpreting-any-machine-learning-model-4f10787ef504#:~:text=Interpretable%20models%20are%20models%20who,vector%20machines%20to%20neural%20networks, (дата звернення: 26.10.2020).

15. Attention для чайников и реализация в Keras. URL: https://habr.com/ru/post/458992/, (дата звернення: 26.10.2020).

16. A Review of Different Interpretation Methods in Deep Learning (Part 1: Saliency Map, CAM, Grad-CAM). URL: https://medium.com/@mrsalehi/a-review-of-different-interpretation-methods-in-deep-learning-part-1-saliency-map-cam-grad-cam-3a34476bc24d, (дата звернення: 26.10.2020).

17. Demystifying Convolutional Neural Networks Using Class Activation Maps. URL: https://towardsdatascience.com/demystifying-convolutional-neural-

networks-using-class-activation-maps-fe94eda4cef1, (дата звернення: 26.10.2020).

18. How do machine learning algorithms work? On the example of LIME and model explainability. URL: https://theblue.ai/blog/lime-models-explanation/, (дата звернення: 26.10.2020).

19. Feature Visualization. URL: https://distill.pub/2017/feature-visualization/, (дата звернення: 26.10.2020).

20. Новічонок М.С., О.В. Яковлева. Розробка програмного застосунку для класифікації об’єктів на зображеннях з використанням бібліотеки TensorFlow. *Інформатика, управління та штучний інтелект*: тези 6 міжнар. наук.-техн. конф. (Харків-Краматорськ, 27-29 листопада 2019 р.). Харків: НТУ «ХПІ», 2019. С. 129. URL: http://pim.net.ua/arch_f/tez_iyii_2019.pdf, (дата звернення: 29.10.2019).

21. Свёрточные нейронные сети: взгляд изнутри. URL: http://ru.datasides.com/code/cnn-convolutional-neural-networks/, (дата звернення: 27.10.2020).

22. А.В.Кухарев, Ю.Н.Махдаев. Нейронные сети на основе операции свертки для эффективного распознавания рукописных цифр. Веснік ВДУ.–2017.–№3(96).С.28-34. URL: https://lib.vsu.by/xmlui/bitstream/handle/123456789/12113/28-34.pdf?sequence=1&isAllowed=y, (дата звернення: 27.10.2020).

23. Функция SoftMax. URL: https://www.asozykin.ru/deep_learning/2018/10/26/Softmax-Function.html, (дата звернення: 27.10.2020).

24. The Functional API. URL: https://keras.io/guides/functional_api/, (дата звернення: 27.10.2020).

25. Новічонок М.С., О.В. Яковлева. Навчені моделі нейронних мереж бібліотеки TensorFlow для рішення задачі класифікації об’єктів на зображенні. *Free and Open Source Software*: тези 11 міжнар. наук.-техн. конф. (Харків, 19-21 листопада 2019 р.). Харків: ХНУБА, 2019. С. 58. URL: https://foss.kn-it.info/uploads/foss-2019-theses.pdf, (дата звернення: 29.10.2019).

26. Архитектура сверточной нейронной сети для классификации типов сцен мобильного робота. URL: https://www.top-technologies.ru/ru/article/view?id=37190, (дата звернення: 27.10.2020).

27. A Review of Popular Deep Learning Architectures: ResNet, InceptionV3, and SqueezeNet. URL: https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/, (дата звернення: 27.10.2020).

28. Rethinking the Inception Architecture for Computer Vision. URL: https://arxiv.org/abs/1512.00567, (дата звернення: 27.10.2020).

29. Batch Normalization In Neural Networks Explained (Algorithm Breakdown). URL: https://towardsdatascience.com/batch-normalization-explained-algorithm-breakdown-23d2794511c, (дата звернення: 08.11.2020).

30. MobileNet: меньше, быстрее, точнее, Habr. URL: https://habr.com/ru/post/352804/, (дата звернення: 27.10.2020).

31. Новічонок М.С., О.В. Яковлева. Аналіз згорткових нейронних мереж бібліотеки TensorFlow для вирішення задачі детектування та класифікації об'єктів на зображенні. *Радіоелектроніка та молодь у XXI столітті*: матеріали 24 молодіжного міжнар. форуму (Харків, 7–9 квітня 2020 р.). Харків: ХНУРЕ, 2019. С. 16–17.

32. Residual blocks – Building blocks of ResNet. URL: https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec, (дата звернення: 11.11.2020).

33. An Overview of ResNet and its Variants. URL: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035, (дата звернення: 11.11.2020).

34. An ResNet (34, 50, 101): «остаточные» CNN для классификации изображений. URL: https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/, (дата звернення: 11.11.2020).

35. Understanding how LIME explains predictions. URL: https://towardsdatascience.com/understanding-how-lime-explains-predictions-d404e5d1829c, (дата звернення: 16.11.2020).

36. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. URL: https://arxiv.org/abs/1602.04938, (дата звернення: 14.11.2020).

37. Keras Applications. URL: https://keras.io/api/applications/, (дата звернення: 15.11.2020).

38. Метрики в задачах машинного обучения. URL: https://habr.com/ru/company/ods/blog/328372/, (дата звернення: 15.11.2020).

39. Жуков Г.П., Викулов С.Ф. Военно-экономический анализ и исследование операций. Москва, 1987. 400 с.

40. Орлов А.И. Высокие статистические технологии: Экспертные оценки : учебник. Москва, 2007. 372 с.