Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-23-1
Шкарупа А.О.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

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

Допускається до захисту

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

2025 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)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шкарупі Альоні Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та реалізація методу класифікації зображень за допомогою нейромережі

затверджена наказом по університету від 25 листопада 2024 року № 1246Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, бібліотека глибокого навчання з відкритим кодом Tensorflow, дані інтернет-мережі, мова програмування Python, середовище розробки Jupyter Notebook на платформі Kaggle, власний набір зображень одягу.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд існуючих методів класифікації зображень та їхньої ролі у сучасних технологіях.
2. Дослідження архітектур згорткових нейронних мереж (CNN), які використовуються для класифікації зображень одягу.
3. Обґрунтування вибору інструментів та програмних засобів для реалізації класифікації.
4. Реалізація п'яти різних архітектур CNN для класифікації зображень одягу.
5. Порівняння точності та ефективності реалізованих моделей у середовищі розробки та у мобільному застосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми класифікації зображень, постановка задачі, схема роботи алгоритму класифікації одягу, тестові зображення, результати роботи класифікації.

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 кваліфікаційну роботу	25.11.2024	
2	Аналіз завдання, підбір літератури	25.11.24-27.11.24	
3	Аналіз літератури з досліджуваної проблеми	27.11.24-28.11.24	
4	Аналіз програмних засобів та інструментів	28.11.24-29.11.24	
5	Розробка методу класифікації зображень	30.11.24-03.12.24	
6	Програмна реалізація	03.12.24-12.12.24	
7	Оформлення пояснювальної записки	12.12.24-17.12.24	
8	Перевірка на плагіат	18.12.2024	
9	Рецензування	21.12.2024	
10	Підготовка презентації та доповіді	23.12.2024	
11	Занесення роботи в електронний архів	01.01.2025	
12	Попередній захист кваліфікаційної роботи	02.01.2025	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 80 с., 1 табл., 57 рис., 50 джерел.

КОМП'ЮТЕРНИЙ ЗІР, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, НЕЙРОННІ МЕРЕЖІ, ОПТИМІЗАЦІЯ МОДЕЛЕЙ, КВАНТУВАННЯ МОДЕЛЕЙ, ТОЧНІСТЬ КЛАСИФІКАЦІЇ, МЕТРИКИ ТОЧНОСТІ.

Об'єктом дослідження є процес класифікації зображень одягу за допомогою нейронних мереж.

Метою дослідження є аналіз і оптимізація процесу класифікації зображень одягу з використанням нейронних мереж, розробка і впровадження різних моделей для цієї задачі, а також оцінка їхньої ефективності на власному наборі даних.

Було використано п'ять архітектур: MobileNetV2, EfficientNetB0, DenseNet121, NASNetMobile та ResNet50V2. Застосовувалися аугментація даних, регуляризація, налаштування гіперпараметрів та оптимізація через Adam. Моделі оцінювалися за допомогою графіків втрат і точності навчання, матриці плутанини та метрик precision, recall, F1-score, ROC-AUC і Log Loss, із візуалізацією результатів для SavedModel, TFLite і Quantized TFLite.

У результаті дослідження розроблено метод класифікації одягу, створено унікальний датасет і мобільний застосунок.

COMPUTER VISION, IMAGE CLASSIFICATION, NEURAL NETWORKS, MODEL OPTIMIZATION, MODEL QUANTIZATION, CLASSIFICATION ACCURACY, ACCURACY METRICS.

The object of the research is the process of classifying clothing images using neural networks.

The aim of the research is to analyze and optimize the process of clothing image classification using neural networks, develop and implement various models for this task, and evaluate their effectiveness on a real-world dataset.

Five architectures were used: MobileNetV2, EfficientNetB0, DenseNet121, NASNetMobile, and ResNet50V2. Data augmentation, regularization, hyperparameter tuning, and Adam optimization were applied. The models were evaluated using training loss and accuracy plots, confusion matrix, and metrics such as precision, recall, F1-score, ROC-AUC, and Log Loss, with visualization of the results for SavedModel, TFLite, and Quantized TFLite.

As a result of the research, a method of clothing classification was developed, a unique dataset and a mobile application were created.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AUC – Area Under Curve (площа під кривою)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

ELU – Exponential Linear Unit (експоненційна лінійна одиниця)

FPR – False Positive Rate (частота хибнопозитивних результатів)

GPU – Graphics Processing Unit (графічні процесори)

HOG – Histogram of Oriented Gradients (гістограма орієнтованих градієнтів)

RGB – Red, Green, Blue (червоний, зелений, синій)

ReLU – Rectified Linear Unit (лінеарно-коригуюча функція)

ROC – Receiver Operating Characteristic (характеристика приймача)

SIFT – Scale-Invariant Feature Transform (масштабно-інваріантне перетворення ознак)

SVM – Support Vector Machines (машина опорних векторів)

TPR – True Positive Rate (частота істиннопозитивних результатів)

ВСТУП

В умовах сучасної інформаційної ери комп’ютерний зір стає все важливішим у різних сферах, таких як наука, промисловість і суспільство. Завдяки впровадженню спеціальних архітектур нейронних мереж, які імітують функціонування людської зорової системи, було досягнуто значних результатів у класифікації зображень та обробці великих обсягів візуальної інформації [1]. Цей прогрес став можливим завдяки революційним можливостям комп’ютерного зору, які справили враження на технологічний світ і суспільство загалом.

Згорткові нейронні мережі створені для виявлення локальних ознак у зображеннях за допомогою згорткових шарів. Спочатку шари знаходять прості контури, а далі з ростом глибини мережі вони починають розпізнавати більш складні структури. Пулінгові шари сприяють зменшенню обсягу даних, що підвищує ефективність обчислень та робить мережу менш чутливою до дрібних змін у зображеннях.

Існує багато моделей нейронних мереж, які демонструють перспективні результати і проблема полягає в тому, що їх ефективність може суттєво варіюватися залежно від специфіки завдання, якості даних і вибору параметрів моделі. Тому актуальність цього дослідження зумовлена потребою в систематичному аналізі, вдосконаленні та порівнянні різних архітектур нейронних мереж для класифікації зображень. Це дозволить не тільки покращити точність класифікації, але й адаптувати ці моделі для реального використання.

Очікується, що результати дослідження покращать якість класифікації одягу, виявлять сильні та слабкі сторони архітектур, а також будуть корисні для реальних застосунків у моді та роздрібній торгівлі, зокрема для інтеграції в мобільні застосунки для класифікації одягу.

1 ОГЛЯД ЛІТЕРАТУРИ ТА СУЧАСНИХ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Класифікація зображень та її роль в сучасних технологіях

Класифікація зображень є дуже важливою складовою сучасних технологій, що значно впливає на різні сфери життя [2], таких як медицина, безпека, автомобільна промисловість, соціальні мережі та роздрібна торгівля. Особливе значення ця задача набуває в контексті моди і роздрібної торгівлі, де точна і швидка класифікація одягу має безпосередній вплив на ефективність бізнес-процесів і задоволеність споживачів.

З розвитком онлайн-шопінгу і персоналізованих сервісів, функціональність систем класифікації зображень набуває все більшої важливості [3]. Сучасні технології дозволяють автоматизувати процеси пошуку товарів, формування рекомендацій та обробки запитів клієнтів, що веде до підвищення ефективності і конкурентоспроможності компаній. Точна класифікація одягу полегшує споживачам процес пошуку необхідних товарів, зменшує час, витрачений на перегляд асортименту, і покращує загальний досвід покупок, що в свою чергу веде до підвищення рівня продажів і лояльності клієнтів [4].

Нейронні мережі, зокрема сучасні архітектури, такі як EfficientNetB0, DenseNet121, MobileNetV2, NASNetMobile і ResNet50V2, демонструють високу ефективність у класифікації зображень [5, 6] завдяки своїй здатності обробляти великі обсяги даних і навчатися на основі чисельних прикладів. Їх застосування в класифікації одягу дозволяє досягти високих результатів у точності розпізнавання різних типів одягу, що є критично важливим для сучасного ринку моди та електронної комерції.

Автоматизація процесів класифікації одягу дозволяє компаніям скоротити витрати на робочу силу та знизити кількість помилок, пов’язаних з

неправильними класифікаціями товарів. Це в свою чергу може призвести до зменшення кількості повернень та обробок скарг клієнтів, що позитивно вплине на репутацію та прибутковість бізнесу.

Ще одним важливим аспектом є масштабованість рішень на основі нейронних мереж до масштабування та їхньої інтеграції з іншими бізнес-системами, такими як рекомендаційні системи або системи управління запасами. Поєднання класифікації зображень з технологіями прогнозування попиту допомагає компаніям ефективніше управляти запасами та покращувати логістичні процеси.

Крім того, точна класифікація одягу сприяє створенню більш персоналізованих рекомендацій, що підвищує якість взаємодії користувачів із сервісом. Це може стимулювати повторні покупки і довгострокову лояльність до бренду.

Проблема полягає в тому, що хоча існує багато моделей нейронних мереж, які демонструють перспективні результати, їх ефективність може суттєво варіюватися залежно від специфіки завдання, якості даних і вибору параметрів моделі [7]. Тому актуальність цього дослідження зумовлена потребою в систематичному аналізі, вдосконаленні та порівнянні різних архітектур нейронних мереж для класифікації зображень одягу. Це дозволить не тільки покращити точність класифікації, але й адаптувати ці моделі для реального використання.

Таким чином, дослідження, яке включає налаштування архітектур нейронних мереж, оптимізацію їх параметрів і порівняння їх продуктивності, має значний потенціал для поліпшення технологій класифікації зображень і може внести важливий вклад у розвиток інновацій у сфері моди і роздрібної торгівлі. Це дослідження дозволить виявити сильні та слабкі сторони існуючих моделей і забезпечить основу для розробки більш ефективних рішень, які можуть позитивно вплинути на конкурентоспроможність і ефективність ринку.

1.2 Ключові завдання, які вирішує класифікація зображень одягу

В сфері комп’ютерного зору класифікація зображень є важливим підходом для аналізу візуальної інформації, що дозволяє системам автоматично визначати до якої категорії належить зображення. Процес класифікації здійснюється за допомогою глибоких нейронних мереж, що навчаються на великих наборах даних. Модель вивчає особливості зображень, що належать до різних класів і на основі цих знань здатна точно класифікувати нові зображення. Кінцева мета полягає в тому, щоб забезпечити високу точність класифікації при роботі з невідомими даними.

До впровадження глибинного навчання, процес класифікації зображень одягу здійснювалося за допомогою традиційних методів машинного навчання [8]. Цей підхід включав використання ручного витягнення ознак, що складалося з кількох етапів. Спочатку зображення аналізувалися з метою виділення їх ключових особливостей, наприклад текстури, ключові точки та контури. Для цього використовували алгоритми SIFT та HOG [9–11].

На наступному етапі отримані ознаки подавалися на вхід класифікаторам, наприклад випадкові ліси або SVM. Ці алгоритми приймали рішення до якого класу відноситься зображення одягу.

Хоча ці методи давали непогані результати при класифікації, наприклад, окремих предметів одягу, але вони мали проблеми з обробкою великих та різноманітних наборів даних. Основна проблема була в тому, що ручне витягування ознак було недостатньо ефективним, а точність класифікації була невисокою. Саме це стало поштовхом до розвитку нових підходів, таких як глибинне навчання, яке значно покращило ефективність і точність процесу класифікації.

Саме глибинне навчання стало ключовою інновацією, яка суттєво змінила підхід до класифікації зображень. Завдяки автоматизації процесу обробки даних, згорткові нейронні мережі замінили ручне витягування ознак на автоматичне навчання моделей. Ці мережі використовують багатошарові

згорткові фільтри, які дозволяють їм самостійно визначати важливі особливості зображень, наприклад контури, форми та текстури, що значно підвищило швидкість та точність класифікації. Це спрощує обробку великих обсягів даних та дозволяє застосовувати моделі в реальних сценаріях, де важлива не лише точність, а й ефективність.

Одним з прикладів використання цих технологій є класифікація одягу. Цей процес включає кілька основних кроків. Спочатку модель визначає області зображення, де розташовані предмети одягу. Потім аналізує їхні характеристики, і після цього система визначає, до якого класу або категорії належить кожен об’єкт. Наприклад, на зображенні можуть бути футболка, штани або сукня, і завдання моделі полягає в тому, щоб правильно класифікувати кожен з цих предметів (рис. 1.1).

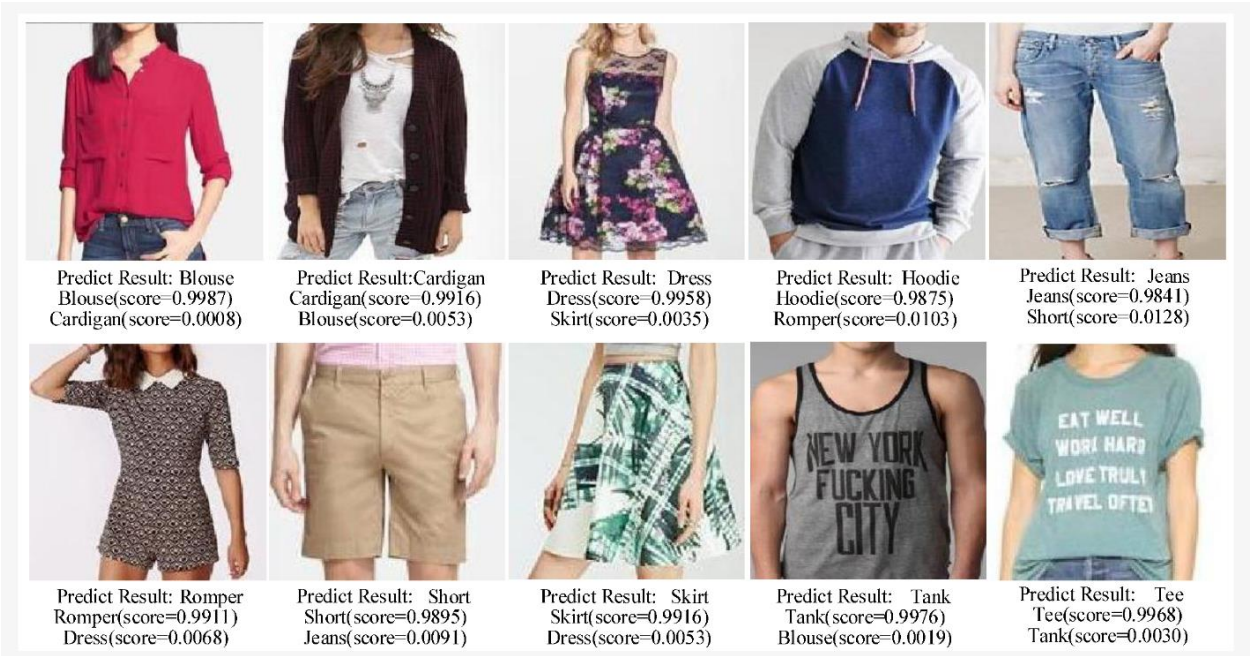

Рисунок 1.1 – Приклад класифікації одягу [12]

Область класифікації зображень стискається з численними викликами, які пов’язанні зі складністю та мінливістю візуальних даних. Основна проблема полягає в тому, що зовнішній вигляд зображень може змінюватися під впливом різних факторів, таких як освітлення, положення, масштаб або

часткове перекриття (оклюзія), що значно впливає на ефективність алгоритмів. Наприклад, одна і та сама футболка може виглядати по-різному залежно від кута зйомки або освітлення (рис. 1.2), що ускладнює навчання моделей глибокого навчання та вимагає аугментації даних.

Рисунок 1.2 – Футболка під різними кутами зйомки та освітлення [12]

Аугментація допомагає розширити кількість тренувальних зображень, застосовуючи різноманітні трансформації [13, 14], такі як перевороти, обертання або зміни яскравості. Це дозволяє зробити модель більш стійкою до різноманітних умов зйомки і зменшити залежність від конкретних налаштувань, але не вирішує проблему повністю. Часткова оклюзія, коли одяг частково закритий іншими об’єктами, наприклад, рукою, сумкою або тінями, може створювати додаткові труднощі для класифікації моделей. На рисунку

1.3 показано зображення з прикладами часткової оклюзії одягу: рука, що прикриває частину одягу, сумка, яка висить на плечі і закриває частину одягу, та тіні, що частково закривають одяг у яскравому світлі.

Рисунок 1.3 – Приклади часткової оклюзії одягу [12]

Іншою важливою проблемою є присутність фонових шумів або додаткових об’єктів на зображенні. На зображеннях окрім одягу, можуть бути люди, меблі або природа, що можуть відволікати модель від основного завдання класифікації. Наприклад, при класифікації зображень футболки модель може помилково звертати увагу на фон або інші об’єкти, а не на сам одяг. На рисунку 1.4 продемонстровано приклад сцени, де фон може відволікати увагу від основного об’єкта футболки. Це може призвести до неправильної класифікації та зменшення загальної точності моделі.

Рисунок 1.4 – Приклад сцени, де фон відволікає увагу від основного об’єкта футболки [12]

Ще одна проблема з якою можна зіткнутися – це масштабованість. Сучасні системи класифікації одягу часто мають справу з величезними наборами даних, які можуть містити мільйони зображень різних категорій. Це створює значне навантаження на обчислювальні ресурси та ускладнює процес навчання моделі [15]. Однак, може бути ще дисбаланс між класами, коли одні категорії одягу мають більше прикладів для навчання, ніж інші, що призводить до зміщення моделі на користь частіших класів. І як наслідок, рідкісні категорії, такі як вечірні сукні або весільні вбрання, можуть класифікуватися з нижчою точністю порівняно з звичайними повсякденними речами, як футболки чи джинси.

Перенесення моделей на раніше не бачені або нові набори даних є також серйозною проблемою у класифікації зображень. Моделі, які демонструють

високу точність на одному наборі даних, можуть не працювати так само добре на інших через відмінності в структурі або якості даних. Це може бути пов’язано з тим, що моделі схильні до того, щоб підлаштовуватися під різні закономірності в тренувальних даних, та коли ці закономірності змінюються або зовсім відсутні в нових наборах, то точність моделі може різко знизитися.

Значною перешкодою на шляху до широкого застосування є також вимоги до пам’яті і обчислювальних ресурсів для навчання та розгортання складних моделей класифікації [16], особливо коли потрібно працювати в умовах обмежених ресурсів або на периферійних пристроях.

Отже, класифікація зображень стискається з низкою складних проблем. Серед них варто відзначити мінливість даних, проблеми масштабованості, труднощі з адаптацією моделей до нових наборів даних та високі обчислювальні вимоги. Ці всі виклики вказують на потребу в удосконаленні існуючих рішень та відкривають нові можливості для інновацій у цій сфері. Також розробка більш стійких до перешкод і менш ресурсозатратних моделей є ще одним напрямом розвитку, що дозволить використовувати класифікацію зображень на різних платформах, включаючи пристрої з обмеженими ресурсами.

1.3 Огляд літератури щодо застосування CNN для класифікації зображень одягу

В даній роботі розглядаються сучасні методи та підходи до класифікації зображень одягу, що є важливим аспектом для застосування в електронній комерції, управлінні запасами, а також для розробки систем комп’ютерного зору у сфері моди.

У джерелі [17] досліджено систему комп’ютерного зору для класифікації зображень одягу. Проведено експерименти з використанням архітектур нейромереж, таких як ResNet та SqueezeNet, а мережу для виявлення одягу

було навчено і протестовано на наборі даних DeepFashion. Задачу класифікації оцінювали на зображеннях суконь з онлайн-магазинів, використовуючи п’ять атрибутів: колір, візерунок, довжина рукава, виріз горловини та лінія подолу. Автоматичне збирання міток дало середню точність 83%. Оцінено можливі покращення, такі як збільшення розміру мережі, використання ансамблів та генерація різноманітних фонових зображень, а також проаналізовано вплив цих покращень на точність класифікації та ефективність обробки. У результаті визначено найбільш успішні конфігурації мереж для класифікації суконь.

У джерелі [18] досліджено метод класифікації силуетів різних видів одягу. Розглянуто два підходи до аналізу одягу: перший фокусується на базових елементах дизайну одягу з точки зору його творця, другий – на детальних категоріях, пов’язаних з виготовленим одягом. Силует одягу є одним з найважливіших аспектів у модних дизайнерських тенденціях. У цьому дослідженні основна увага приділяється класифікації силуетів, що сприяють створенню тенденцій, а не класифікації самих предметів одягу. Ця класифікація дозволяє створити набір даних силуетів, який може бути використаний для багатокласової класифікації в глибокій нейронній мережі.

У джерелі [19] досліджено алгоритм класифікації зображень одягу на основі покращеної моделі Xception. Було здійснено кілька покращень: заміна останнього повнозв’язного шару для розпізнавання восьми класів замість 1000, використання активаційних функцій ELU і ReLU для покращення нелінійних і навчальних характеристик мережі, застосування L2-регуляризації для підвищення стійкості мережі до перешкод, а також збільшення даних для тренувальних зображень для зменшення перенавчання. Крім того, було налаштовано навчання мережі шляхом фіксації швидкості навчання в перших двох модулях. Експериментальні результати показали, що запропонований алгоритм досягає точності на рівні 92%, що перевершує сучасні моделі, такі як Inception-v3, Inception-ResNet-v2 і Xception.

У джерелі [20] досліджено інтегровану систему для спільного парсингу зображень одягу, яка дозволяє розбивати набір зображень одягу на семантичні

конфігурації. Запропоновано нову систему на основі даних, що складається з двох етапів інференції. На першому етапі, який називається «Спільна сегментація зображень», виділяються узгоджені області на зображеннях і спільно уточнюються ці області за допомогою техніки exemplar-SVM. На другому етапі «Спільне маркування областей», що створюється графічна модель для декількох зображень, де сегментовані області є вершинами, і враховуються різні контексти конфігурацій одягу. Наприклад, розташування предметів та взаємодія між ними. Спільне призначення міток вирішується ефективним алгоритмом Graph Cuts. Система протестована на наборі даних Fashionista та новому наборі SYSU-Clothes, який містить 2098 фотографій вуличної моди високої роздільної здатності. Досягнута точність сегментації 88%–90% та точність розпізнавання 63%–65% на наборах Fashionista та SYSU-Clothes, що перевершує попередні методи.

У джерелі [21] досліджено техніку класифікації зображень на основі покращеної CNN моделі з удосконаленими згортковими та пулінговими шарами. Додатково було застосовано алгоритм динамічного регулювання швидкості навчання, який дає змогу адаптивно змінювати її, прискорюючи збіжність моделі та зменшуючи тривалість тренувального процесу.

Експеримент з використанням запропонованої моделі було проведено на датасеті Fashion-MNIST, де для навчання використали 6000 зображень, а для тестування 1000 зображень. Запропонований метод показав точність класифікації 93%, що перевищує базову CNN модель на 4%. Також було проведено аналіз впливу розміру пакету на точність. Результати експериментів продемонстрували, що модель добре узагальнює задачі класифікації зображень одягу.

У джерелі [22] розглядається застосування згорткових нейронних мереж для класифікації зображень одягу на основі таких метрик, як F1-score, точність, precision та recall. Основною метою дослідження було підвищення точності розпізнавання та класифікації зображень для допомоги в підборі одягу, його повторній обробці, а також для оптимізації управління запасами у

роздрібній торгівлі. Було проведено порівняння чотирьох моделей із заданими порогами для recall 74% та precision 89%. Модель ZF-NET показала найвищу точність 81%, що свідчить про її високу ефективність у класифікації одягу. Модель VGG-16 досягла точності 86% та продемонструвала хороший баланс між recall та precision. Крім того, була запропонована вдосконалена модель Clothes-DAT, яку порівняли з чотирма іншими моделями за метриками recall, точність та F1-score. Clothes-DAT показала високу здатність до узагальнення та ефективну роботу з начальними і новими даними.

У джерелі [23] розглядається модель глибокого навчання, яка здатна розпізнавати верхню та нижню частини одягу на модних зображеннях, класифікуючи кожен предмет на його матеріальні характеристики. Для цього використовували моделі ResNet та EfficientNet, які були навчені на датасеті з 1002718 зображень та 125 мітками, які включали категорії одягу та його матеріали. За результатами, модель ResNet досягла зваженого F1-Score 0,801, а модель EfficientNet – 0,781. Отже, модель ResNet продемонструвала кращий результат, ніж модель EfficientNet.

1.4 Постановка задачі дослідження

Огляд наукових джерел показав, що тема класифікації зображень одягу активно досліджується, і існує багато різних підходів та методів для вирішення цього завдання.

Однак, існують певні недоліки в існуючих дослідженнях. По-перше, багато робіт зосереджені на використанні стандартних наборів даних і не приділяють достатньої уваги тестуванню моделей на реальних або спеціалізованих даних. Це може обмежити загальну ефективність і узагальнювальність моделей у практичних умовах. По-друге, деякі дослідження не враховують специфіку мобільних платформ, що є важливим аспектом для застосування в реальних мобільних застосунках. По-третє, часто

недостатньо порівнюються різні архітектури нейронних мереж в контексті конкретних завдань класифікації одягу, що ускладнює вибір найбільш ефективної моделі для певного випадку. Ці аспекти потребують удосконалення для досягнення кращих результатів у класифікації зображень одягу.

Об’єктом дослідження є процес класифікації зображень одягу за допомогою нейронних мереж.

Метою дослідження є аналіз і оптимізація процесу класифікації зображень одягу з використанням нейронних мереж, розробка і впровадження різних моделей для цієї задачі, а також оцінка їхньої ефективності на власному наборі даних.

Для досягнення зазначеної мети потрібно вирішити наступні завдання:

– зробити огляд наукових джерел і сучасних методів класифікації зображень одягу, а також виявити основні проблеми та обмеження цих підходів;

– дослідити ключові етапи процесу класифікації зображень одягу за допомогою нейронних мереж із використанням підходу Transfer Learning та оптимізатора Adam;

– проаналізувати актуальні архітектури згорткових нейронних мереж для класифікації одягу, таких як MobileNetV2, EfficientNetB0, DenseNet121, NASNetMobile та ResNet50V2;

– створити власний датасет, який містить різні типи одягу, максимально наближені до практичного використання;

– виконати налаштування та здійснити навчання вибраних архітектур;

– провести квантування і оптимізацію роботи моделей;

– провести оцінку ефективності моделей на основі графіків втрат і точності навчання, матриці плутанини, а також метрик, таких як accuracy, precision, recall, F1-score, ROC-AUC і Log Loss;

– візуалізувати результати класифікації зображень одягу та виконати порівняльний аналіз моделей.

2 ЗАСТОСУВАННЯ НЕЙРОМЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

2.1 Ключові етапи процесу класифікації зображень у CNN

Згорткові нейронні мережі (CNN) – штучні інтелектуальні системи, які побудовані на основі багатошарових нейронних мереж та які здатні розпізнавати та класифікувати об’єкти на зображеннях, а також визначати їх межі. Фактично, CNN є популярною архітектурою в глибокому навчанні, що дозволяє системі навчатися безпосередньо на основі вхідних даних, без необхідності попереднього вилучення людських ознак [24–26].

Важливо глибоко розуміти компоненти згорткових нейронних мереж, такі як згорткові шари, шари об’єднання, шари активації та повнозв’язні шари, які виконують важливу роль в процесах навчання мережі та класифікації зображень. На рисунку 2.1 представлено кілька основних елементів згорткової нейронної мережі, що ілюструють їх взаємозв’язок та вплив на загальний процес обробки вхідних даних.

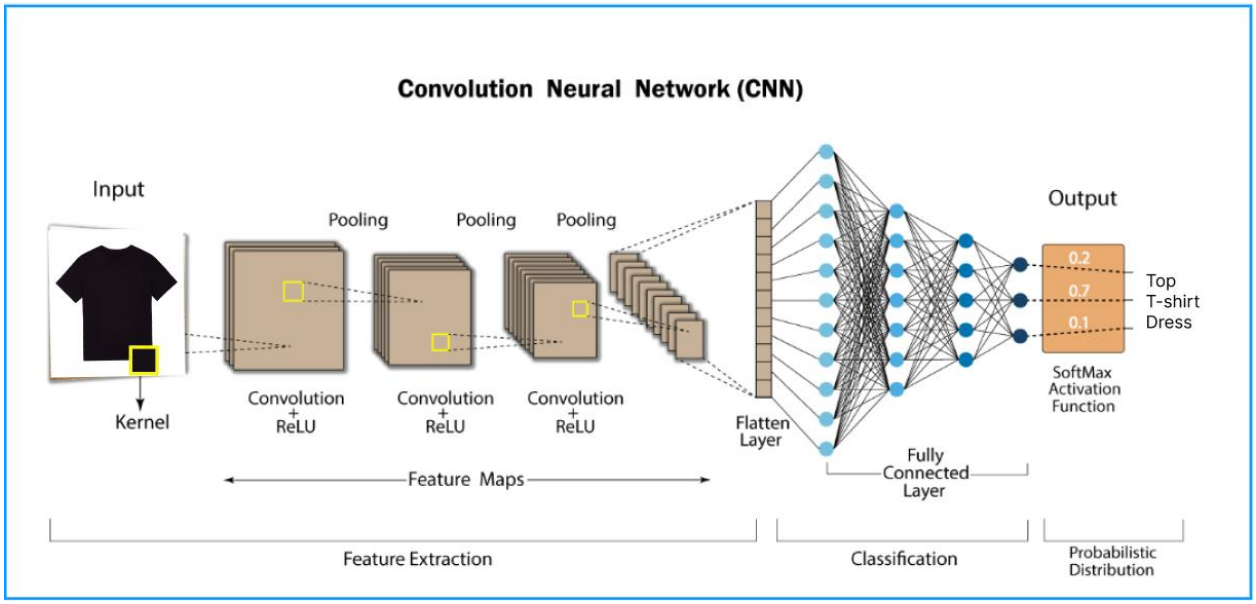

Рисунок 2.1 – Основні компоненти згорткової нейронної мережі

Перший етап згорткової нейронної мережі – введення зображення. Цей етап є критично важливим, бо саме з цього моменту починається процес обробки візуальної інформації. На цьому етапі до моделі передається цифрове зображення у вигляді матриці пікселів, кожен з яких має певне значення, що відповідає інтенсивності яскравості або кольору. Якщо зображення чорно-біле, то значення пікселів знаходяться в діапазоні від 0 до 255, де 0 відповідає чорному кольору, а 255 – білому. Якщо зображення кольорове, то використовується система RGB, де кожен піксель описується трьома компонентами, що визначають інтенсивність кожного з кольорів [27].

Після введення зображення модель переходить до другого етапу – використання шару згортки, що є ключовим компонентом структури згорткової нейронної мережі. На цьому етапі застосовуються фільтри або ядра до вхідних даних, щоб виділити важливі характеристики. Кожне ядро має певні параметри, такі як висота, ширина та ваги, які відповідають за витягування інформації з пікселів зображення. Спочатку ці ваги призначаються випадково, але потім в процесі навчання вони налаштовуються, поступово стаючи більш точними та ефективними для виявлення важливих ознак [28].

Інакше кажучи, карта ознак створюється в результаті комбінування вхідного зображення, яке представлено N-вимірними метриками, з цими фільтрами. Ядро складається з набору числових значень, які використовуються для вилучення характеристик із вхідних даних. Під час початкової ініціалізації ваги ядра задаються випадковим чином, але в процесі навчання вони поступово коригуються для покращення результатів.

Ядро дає можливість виконувати операції в багатовимірному просторі ознак, не потребуючи прямого обчислення координат даних у цьому просторі. Замість цього обчислюється скалярний добуток між фільтрами та вхідними даними, що допомагає виявляти складні закономірності в зображеннях. Завдяки цьому підходу лінійні моделі перетворюються на нелінійні, що робить CNN дуже ефективними для обробки візуальної інформації.

Спочатку подається формат вхідних даних для CNN, а потім починається процес згортки. Звичайна нейронна мережа працює з даними у векторному форматі, тоді як CNN обробляє зображення, що складаються з кількох каналів. Наприклад, RGB-зображення має три канали, кожен з яких відповідає за один із кольорів, тоді як чорно-біле зображення містить лише один канал [28].

Для прикладу, візьмемо градаційне зображення розміром 4×4 з ядром 2×2, яке було ініціалізоване випадковими значеннями, щоб продемонструвати, як працює згортка. Ядро поступово переміщується по зображенню горизонтально і вертикально. Під час цього обчислюється скалярний добуток між ядром та відповідними ділянками зображення. Це здійснюється шляхом множення відповідних значень пікселів на елементи ядра, а потім підсумовування результатів для отримання одного скалярного значення. Операція повторюється доти, доки ядро не охопить всю площу зображення.

Основне зображення та фільтр формують матрицю результату на основі рівняння:

$$(K - L + 1), \tag{2.1}$$

де K – розмір сторони основного зображення;

L – розмір сторони фільтра або ядра.

Для нашого прикладу $4 - 2 + 1 = 3$, що дає результат у вигляді матриці 3×3.

Значення скалярного добутку фактично визначають функціональну карту вихідних даних. На рисунку 2.2 візуально представлено основні обчислення, що відбуваються на кожному етапі. На рисунку менший квадрат розміром 2×2 представляє ядро, а більший квадрат розміром 4×4 – вхідне зображення. Після множення відповідних елементів обох квадратів

результатом є одне число, яке потім використовується як вхідне значення для вихідної карти ознак.

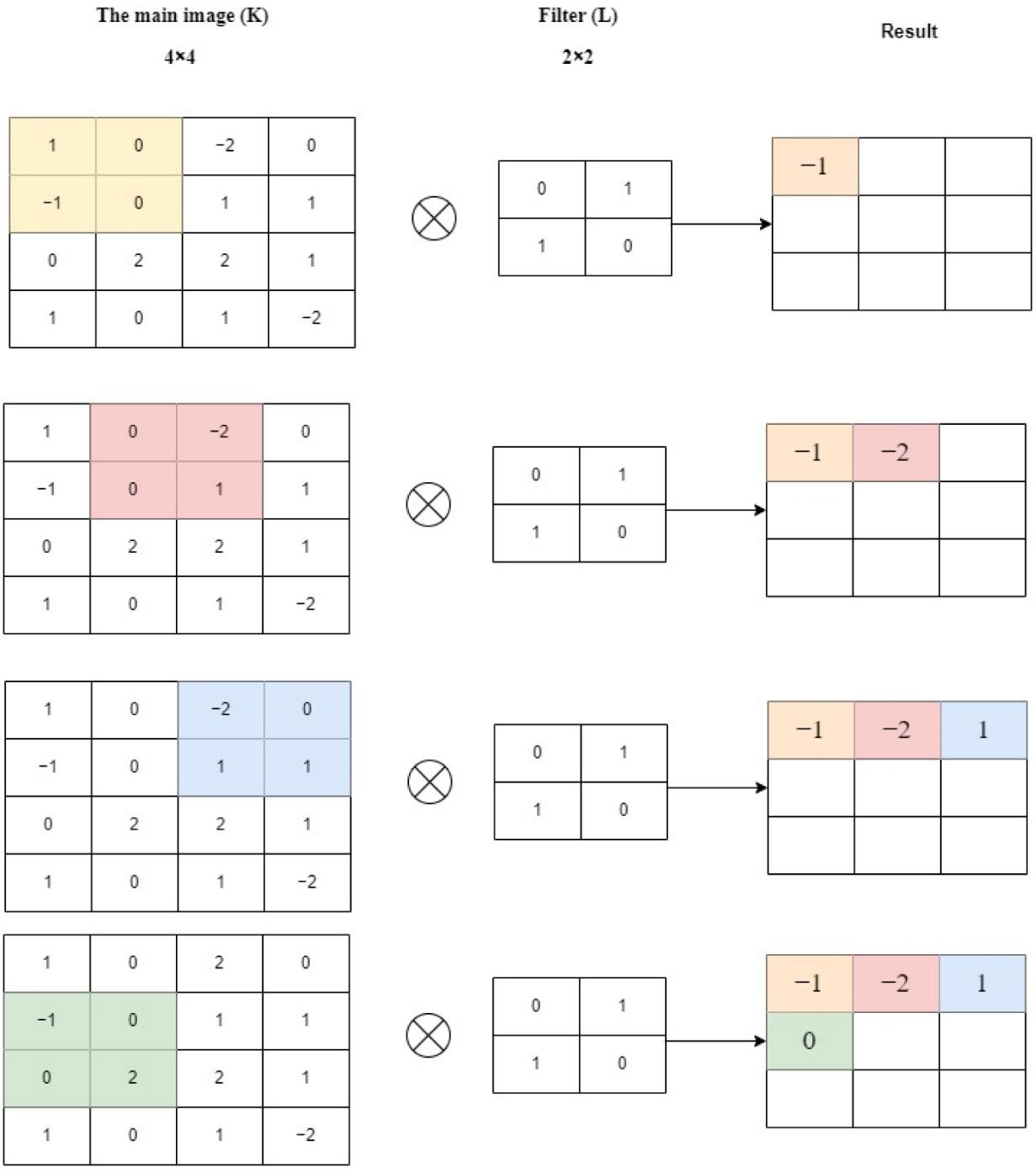

Рисунок 2.2 – Візуальне представлення основних розрахунків

В даному прикладі для ядра було встановлено крок 1, що визначає на скільки пікселів пересувається ядро при обробці зображення по вертикалі або

горизонталі, проте вхідне зображення не було доповнене. Якщо потрібно, то можна змінити величину кроку. Збільшення кроку дозволить зменшити розмір отриманої карти ознак. Доповнення може впливати на розмір меж вхідного зображення. Його застосування збільшує розмір зображення та розмір карти ознак [28].

Кожен фільтр у CNN призначений для виявлення певної ознаки. Якщо фільтр пересувається по зображенню та не знаходить відповідної ознаки, то він не активується. Такий підхід дозволяє CNN визначати найбільш відповідні фільтри для опису об’єктів.

На рисунку 2.3 показано, як за допомогою матриці можна виявляти краї зображення. Через те, що ці матриці виконують роль фільтрів, нагадують звичайні фільтри, що застосовуються в методах обробки зображень.

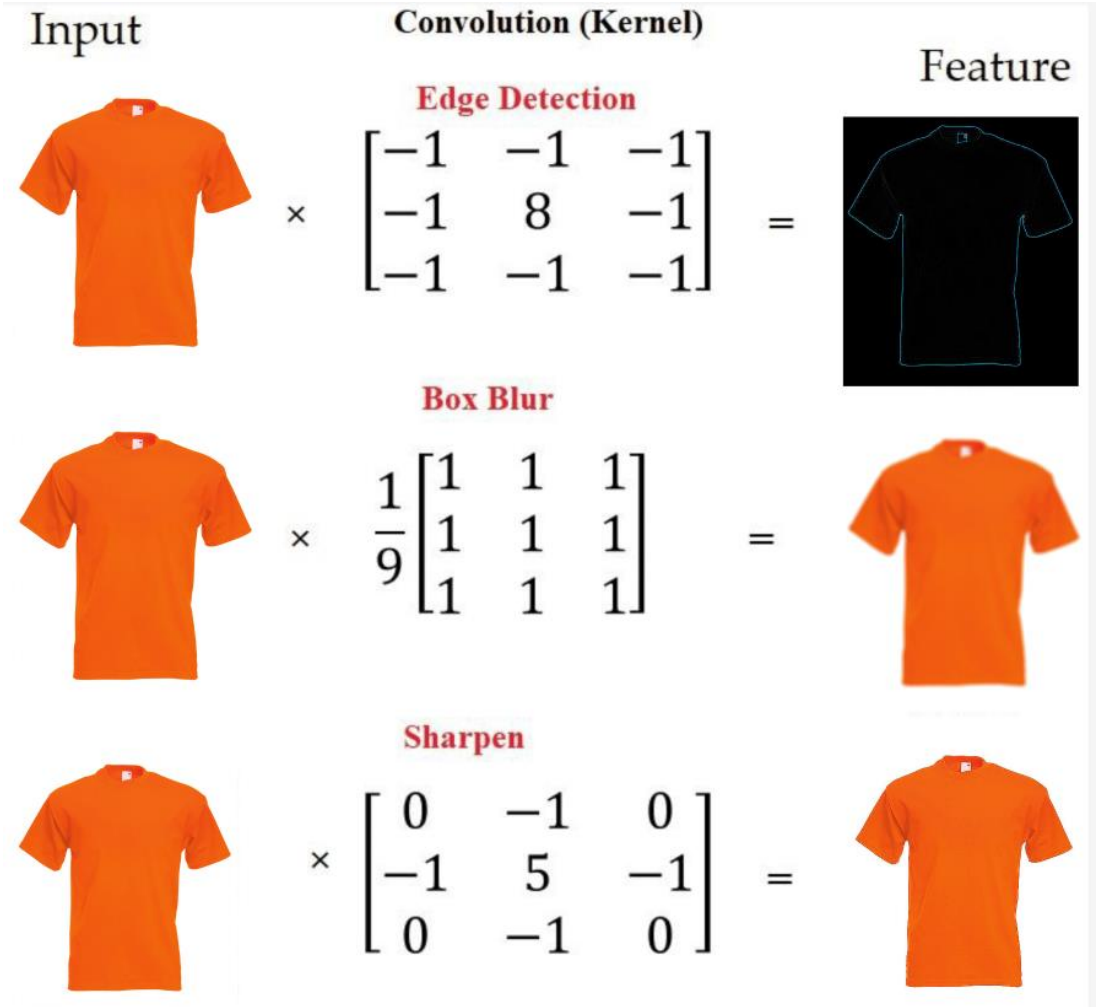

Рисунок 2.3 – Приклад впливу різних матриць згортки

Всі ваги в CNN застосовуються до кожного пікселя вхідного зображення, тому немає окремо призначених ваг між нейронами сусідніх шарів. Це дозволяє навчати одну групу ваг для всієї вхідної інформації, що значно зменшує час навчання та знижує ресурси, адже не потрібно навчати окремі ваги для кожного нейрона [29].

CNN пропонує додаткові можливості для налаштування, що дозволить зменшити небажані ефекти. Одним з таких параметрів є крок (stride). У цьому випадку вихідні вузли наступного шару зазвичай перекриваються зі своїми сусідами на основі аналізу областей. Можна зменшити це перекриття, змінюючи крок. Наприклад, якщо використовувати зображення розміром 6×6, а фільтр переміщувати на одну клітину за раз, то максимальний розмір виходу буде 4×4 (рис. 2.4). Якщо крок збільшити до 2, то вихідна матриця буде 3×3 та перекриття між елементами зменшиться.

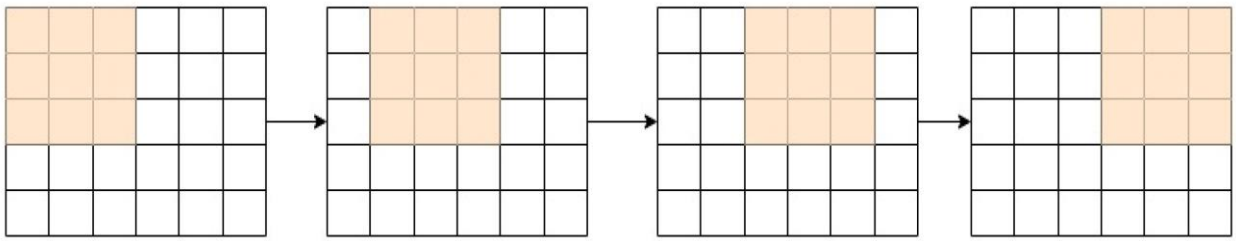

Рисунок 2.4 – Приклад кроку 1, вікна фільтрів переміщуються один раз для кожного з’єднання

Формула (2.2) формалізує це, в результаті отримаємо вихідний розмір O, як показано на рисунку 2.5, враховуючи розмір зображення $N \times N$ та розмір фільтра $F \times F$ [30].

$$O = 1 + (N - F)/S, \tag{2.2}$$

де N – розмір вхідних даних;

F – розмір фільтра;

S – розмір кроку.

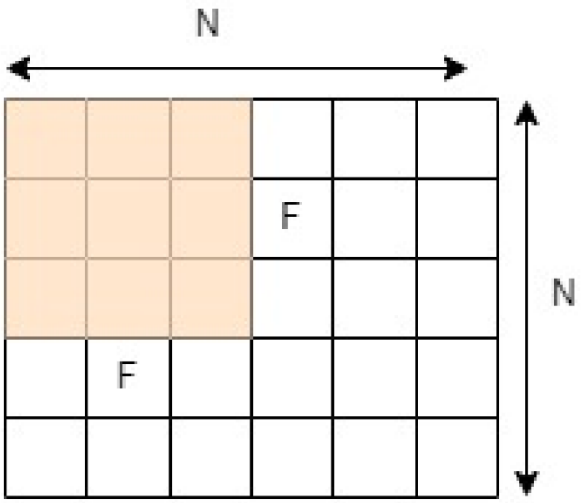

Рисунок 2.5 – Вплив кроку на результат

Під час згортки може виникати проблема втрати інформації на краях зображення, оскільки ці області не завжди потрапляють у поле зору фільтра під час його переміщення. Одним із простих рішень цієї проблеми є використання нульового паддінгу, коли навколо зображення додається рамка з нулів. Це також дозволяє зберегти розмір вихідного зображення або змінити його за потреби.

Наприклад, на рисунку 2.6 вхідне зображення має розмір 6×6, крок дорівнює 1, фільтр $F = 3$, то без паддінгу вихід буде мати розмір 4×4. Однак, якщо додати один шар нульового паддінгу, вихідне зображення залишиться 6×6, як і початковий розмір [30]. У такому випадку формула (2.2) змінюється для випадку з нульовим паддінгом:

$$O = 1 + \frac{N+2P-F}{S}, \tag{2.3}$$

де P – кількість шарів нульового заповнення.

У CNN ваги розподіляються таким чином, що модель залишається стійкою до змін зміщення об’єктів на зображенні, тобто присутня

інваріантність до переміщень. Процес навчання можна налаштувати під будь-яке середовище. Якщо фільтри спочатку мають випадкові значення, то в процесі навчання вони зможуть навчитися розпізнавати краї зображення (рис. 2.2). Однак варто зазначити, що спільні ваги не завжди є ефективними, коли потрібно враховувати просторову важливість вхідних даних [30].

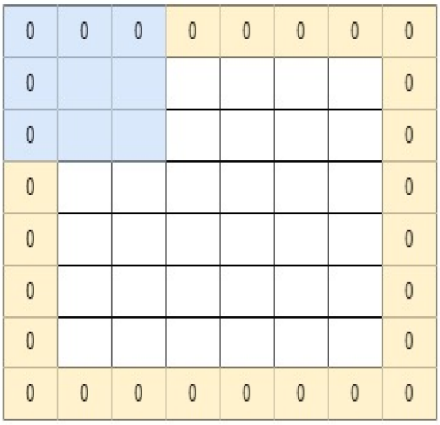

Рисунок 2.6 – Приклад нульового заповнення

Третій етап – це шар пулингу, відомий як шар зменшення вибірки. Він потрібен для того, щоб скоротити розмір карт ознак, зберігаючи при цьому інформацію. У цьому шарі фільтр проходить по вхідних даних та виконує операцію пулингу. Наприклад, вибір мінімального, максимального або середнього значення. Найчастіше застосовується максимальний пулинг.

Основна задача пулингу – це спрощення роботи наступних шарів, завдяки зменшенню обсягу даних. У контексті обробки зображень це може бути зниження роздільної здатності фотографії. Кількість фільтрів при цьому не змінюється. Максимальний пулинг (max-pooling) є одним із найпоширеніших методів. Він працює так: зображення поділяється на невеликі прямокутні області та з кожної області вибирається лише найбільше значення. Найчастіше для цього використовують розмір 2×2 [31].

Як показано на рисунку 2.7, коли пулинг виконується на блоках розміром 2×2 у верхньому лівому куті, фокус зміщується до верхнього правого кута, при цьому здійснюються два кроки. Тобто, для пулингу

використовується крок 2. Однак можна також застосувати крок 1. Це є рідкісною практикою, але її використовують, щоб уникнути зменшення розмірності даних. Також важливо зазначити, що під час зменшення вибірки початкове положення даних втрачається [31].

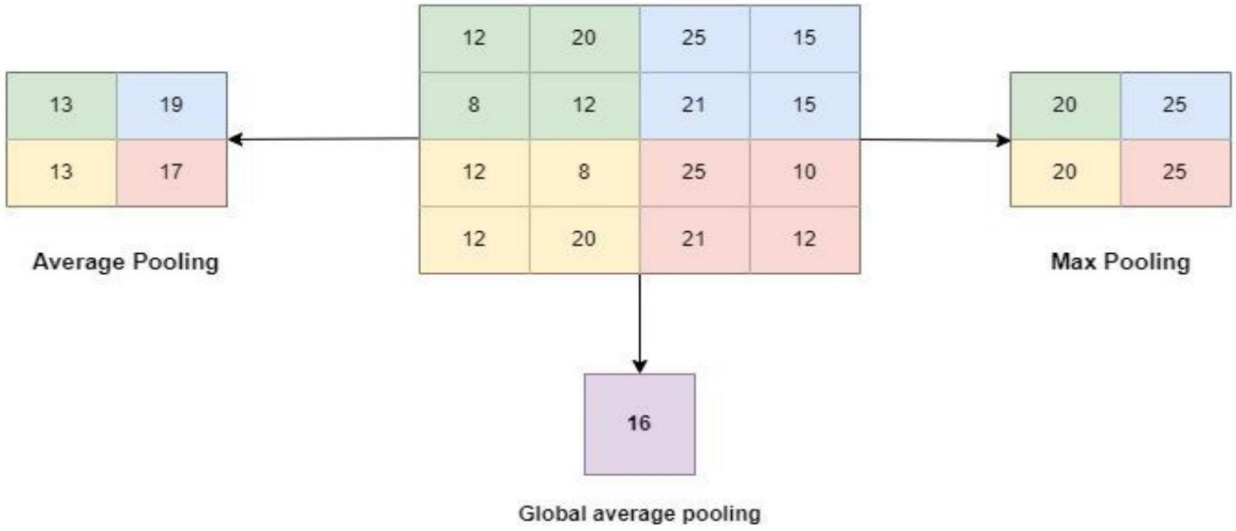

Рисунок 2.7 – Приклад шару пулингу

На різних рівнях пулингу можуть використовуватися різні методи, такі як глобальний максимальний пулинг, глобальний середній пулинг, середній, мінімальний пулинг та пулинг із використанням затворів. На рисунку 2.7 представлені три з цих технік пулингу [31].

Основний недолік шару пулингу полягає в тому, що він не допомагає згортковій нейронній мережі визначати наявність ознаки на вхідному зображенні, а лише показує її місцезнаходження. Через це іноді загальна точність мережі може знижуватись.

Четвертий етап – це нелінійність, функція активації. Цей етап дозволяє змінювати або блокувати вихідний сигнал, обмежуючи або перенасичуючи їх.

Функція активації в нейронних мережах виконує важливу роль у тому, щоб перетворювати вхідні дані у відповідний вихід. Вхід обчислюється як зважена сума сигналів, що надходять до нейрона та його зміщення, якщо воно

є. Функція активації визначає, чи буде нейрон активований у відповідь на цей вхід, генеруючи відповідний вихід [32].

До найпоширеніших функцій активації, які використовуються в CNN та інших глибоких нейромережах включають кілька ключових варіантів (рис. 2.8). Функція Sigmoid приймає на вхід дійсні числа та обмежує вихідні значення в діапазоні від 0 до 1. Функція Tanh подібна до функції Sigmoid, обробляє дійсні числа на вході, але її вихідні значення знаходяться в межах від -1 до 1. Функція ReLU перетворює всі від'ємні вхідні значення на нулі, залишаючи тільки додатні, що робить її більш ефективною в плані використання обчислювальних ресурсів.

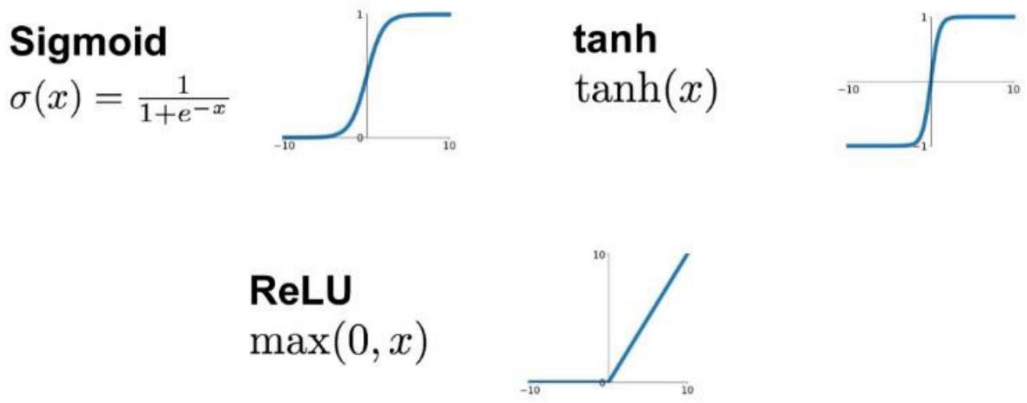

Рисунок 2.8 – Приклади функції активації

Функції сигмоїдна та гіперболічний тангенс мають проблеми зі зворотним поширенням помилки, коли сигнал градієнта поступово слабшає зі збільшенням глибини нейронної мережі. Це відбувається через те, що градієнт цих функцій фактично дорівнює нулю по обидва боки від центру. На відміну від них, у функції ReLU градієнт для додатних вхідних значень є постійним, що вирішує цю проблему. Хоча ReLU не є диференційованою на всьому діапазоні, це не знижує її ефективності в реалізації [32].

Функція ReLU створює більш розріджене представлення, оскільки при нульовому градієнті її вихід також дорівнює нулю. У випадку з функціями Sigmoid і Tanh градієнт ніколи не досягає нуля, що може уповільнити процес

навчання. Однак, у функції ReLU іноді виникає проблема, коли нейрон перестає активуватися через великий градієнт під час зворотного поширення помилки. Для вирішення цієї проблеми були створені альтернативи, наприклад, Leaky ReLU. Ця функція не відкидає повністю негативні значення, а лише зменшує їх [32].

П’ятий та кінцевий етап включає повністю підключений шар, де нейрони організовані в групи, подібні до тих, що зустрічаються в класичних нейронних мережах (рис. 2.9). Кожен вузол цього шару з’єднаний з усіма вузлами в шарах вище та нижче. На рисунку видно, що кожен вузол у нових картах шару пулингу з’єднаний з верхнім шаром через вектор з повністю підключеного шару.

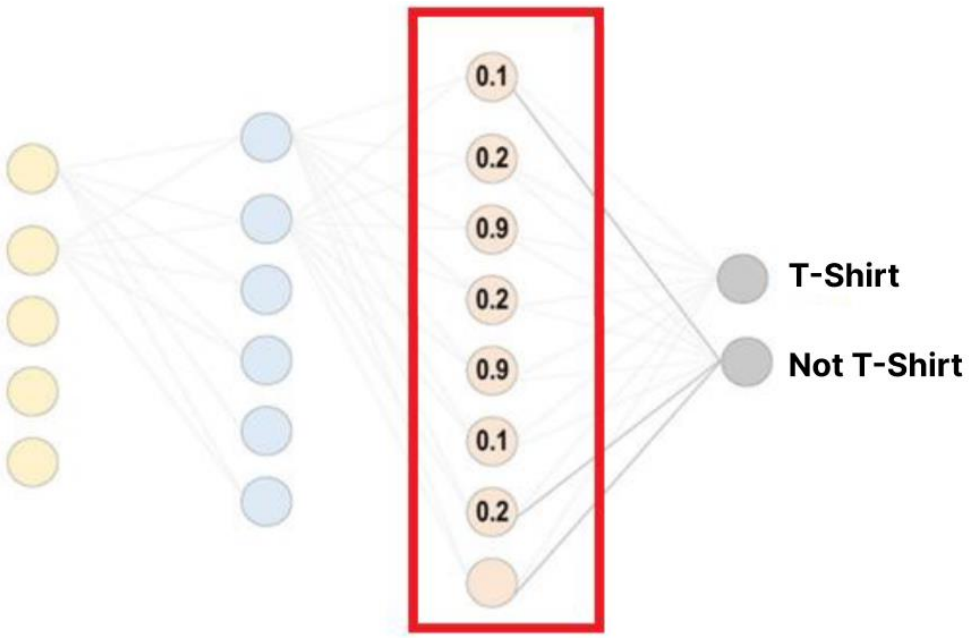

Рисунок 2.9 – Приклад повністю підключеного шару

Однією з головних проблем повністю підключеного шару є велика кількість параметрів, які потребують значних обчислювальних ресурсів під час навчання. Щоб оптимізувати процес, часто зменшують кількість з’єднань і нейронів. Для цього застосовують метод dropout, який дозволяє «відсіяти» частину нейронів і з’єднань, не втрачаючи важливої інформації [32]. На

рисунку 2.9 показано, що вихідний результат повністю підключеного шару узгоджується з кінцевим виходом CNN.

Кінцева класифікація в CNN здійснюється через вихідний шар, який є останнім елементом архітектури мережі. У цьому шарі використовуються різні види функції втрат для обчислення помилок на навчальних даних. Ці помилки показують різницю між фактичним результатом і прогнозованим виходом, а потім ця різниця коригується під час навчання мережі.

Функція втрат використовує два параметри для визначення джерела помилки. Першим є прогнозований або оцінений результат моделі, а другим – бажаний результат або мітка.

Таким чином, було представлено загальну архітектуру згорткової нейронної мережі, де кожен етап виконує свою роль в процесі класифікації зображень. Завдяки цьому підходу CNN здатні ефективно навчатися на великих обсягах даних та демонструвати високі показники точності в задачах класифікації.

2.2 Застосування Transfer Learning та алгоритму Adam для оптимізації CNN

В багатьох ситуаціях навчання моделей з нуля потребує значних ресурсів, особливо коли доступно обмежена кількість даних. Одним із підходів для вирішення цієї проблеми – це застосування Transfer Learning. Цей метод дає змогу використовувати вже навчені моделі, які були вже натреновані на великому наборі даних, для вирішення схожих завдань на нових даних. Завдяки цьому підходу зменшується час навчання та покращується точність моделі.

Процес Transfer Learning (рис. 2.10) починається з того, що використовують модель, яка вже була навчена на великому наборі даних для вирішення схожої задачі. Як правило, такі моделі тренуються на наборі даних

ImageNet, і можуть витягувати важливі характеристики зображень. При цьому самі початкові шари цих моделей, які відповідають за розпізнавання базових ознак, наприклад, кольорових градієнтів або країв, залишаються корисними для нових завдань, які можуть відрізнятися від оригінального завдання [33].

Зазвичай, на етапі перенесення знань початкові шари моделі залишаються «замороженими». Тобто їхні ваги не змінюються під час навчання на новому наборі даних. Це дає змогу зберегти накопичені знання, які були отримані в ході первинного навчання на великому наборі даних. Якщо ці шари вже достатньо добре навчені для розпізнавання базових ознак, то їхнє перенавчання не є необхідним.

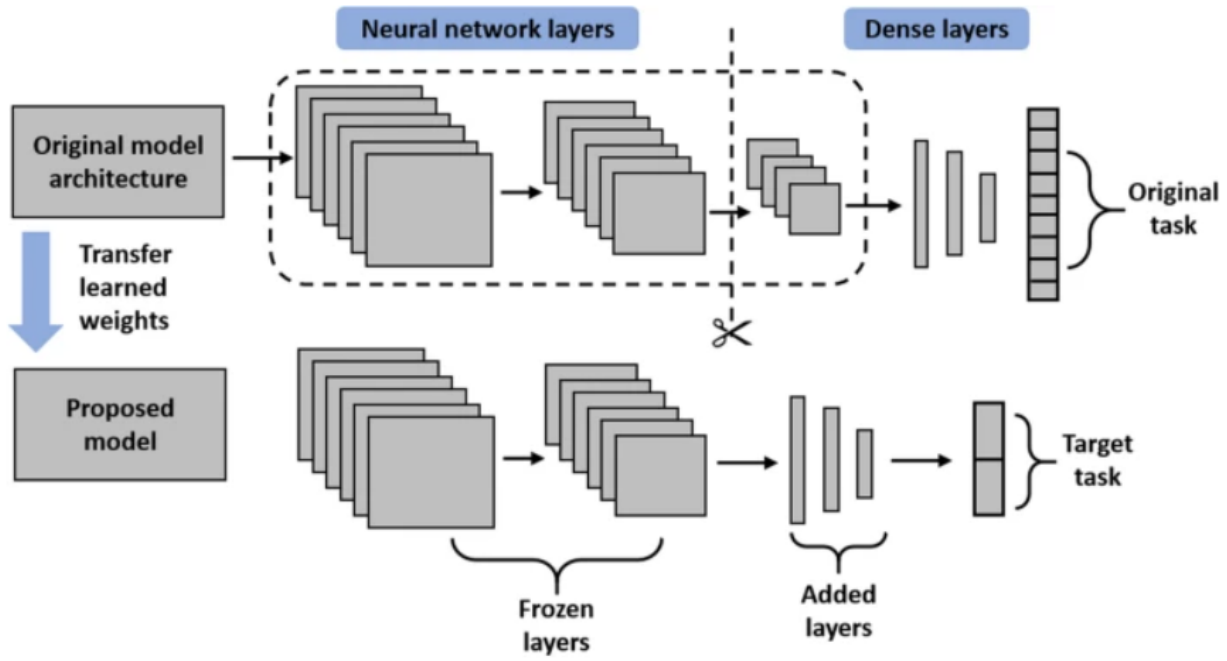

Рисунок 2.10 – Застосування Transfer Learning

Наступним етапом є додавання нових шарів до моделі. Ці шари часто є повнозв’язними або спеціально налаштованими для виконання конкретного завдання. Вони відповідають за класифікацію або інші обчислювальні операції на новому наборі даних. Завдяки тому, що початкові шари моделі вже були навчені витягувати важливі ознаки, то подальше навчання відбувається

швидше та ефективніше. Тобто, модель не починає все з нуля, а лише інтегрує нові знання, які стосуються конкретної задачі [33, 34].

Під час етапу перенавчання відбувається коригування тільки нових шарів, тоді як базові шари, які відповідають за витяг ознак, залишаються без змін. Це значно зменшує час навчання та знижує потребу в обчислювальних ресурсах. В результаті, модель швидко адаптується до нових завдань, використовуючи вже накопичені знання. Наприклад, модель, яка спочатку навчалася розпізнавати автомобілі, може легко бути адаптована до класифікації інших об’єктів, таких як тварини або одяг.

Transfer Learning має багато переваг, але він може виявитися малоефективним, якщо початкова задача сильно відрізняється від нової, оскільки ознаки можуть не відповідати новій задачі. Також існує ризик перенавчання, якщо обсяг нових даних занадто малий. Крім того, даний метод вимагає наявності якісної початкової моделі, а адаптація та підлаштування параметрів можуть зайняти додатковий час [35].

Після успішного застосування Transfer Learning, важливим етапом для підвищення ефективності навчання моделі є вибір правильного алгоритму оптимізації. Одним із дієвих та популярних алгоритмів є оптимізатор Adam. Його основна ідея полягає в тому, щоб для кожного параметра моделі окремо змінювати швидкість навчання, враховуючи як середнє значення градієнтів, так і їхню дисперсію [36, 37].

Алгоритм Adam використовує метод моментуму, який допомагає згладжувати коливання під час навчання моделі. Це означає, що Adam накопичує інформацію про попередні зміни параметрів та враховує її при виборі напрямку оновлення, завдяки чому навчання відбувається швидше та стабільніше. Крім того, алгоритм застосовує адаптивну зміну швидкості навчання для кожного параметра, що може дозволяти враховувати різні масштаби градієнтів та запобігати занадто великим або малим змінам у параметрах.

Ще однією важливою особливістю Adam є виправлення початкових похибок, які можуть виникати у градієнтах на початку навчання. Завдяки цьому алгоритм забезпечує більш точний та стабільний процес налаштування параметрів моделі [36].

Алгоритм виконується в кілька етапів. Спочатку Adam обчислює середнє значення градієнтів та середнє значення їх квадратів. Далі вносить необхідні корегування для уникнення зміщень. Потім оновлює параметри моделі з урахуванням адаптованих значень швидкості навчання.

Однією з основних переваг алгоритму Adam є його здатність автоматично налаштовувати швидкість навчання для кожного параметра окремо, що робить цей метод дуже ефективним при роботі з моделями з великою кількістю параметрів. Завдяки використанню моментуму, Adam дозволяє швидко досягати результатів, скорочуючи час навчання. Крім того, він добре працює на великих обсягах даних та може впоратися з шумом в цих даних, оскільки Adam адаптується до змін градієнтів у процесі навчання [37].

Однак, як і інші алгоритми, Adam має свої недоліки. Він може бути чутливим до налаштувань гіперпараметрів, таких як швидкість навчання та моментум [37]. У деяких випадках, особливо при невеликій кількості даних, алгоритм може швидко знижувати втрати, але це не завжди означає, що знайдене рішення є оптимальним, і може призвести до перенавчання моделі.

Отже, метод Transfer Learning прискорює навчання моделей, використовуючи попередньо натреновані моделі для нових завдань, що знижує потребу в даних і ресурсах. Це дозволяє досягати високої точності навіть на маленьких наборах даних. Алгоритм Adam забезпечує ефективну оптимізацію завдяки адаптивному регулюванню швидкості навчання, однак може бути чутливим до налаштувань гіперпараметрів, що підвищує ризик перенавчання на малих вибірках даних. Тому важливо контролювати процес навчання. Такий підхід особливо корисний для задач, де доступ до великих обсягів даних обмежений.

2.3 Ефективність та особливості сучасних п’яти архітектур нейронних мереж

Модель MobileNetV2 – це легка архітектура згорткових нейронних мереж, яка була розроблена для роботи на вбудованих та мобільних пристроях. Основна мета цієї архітектури полягає в досягненні високої продуктивності при мінімальних витрат обчислення. MobileNetV2 (рис. 2.11) використовує глибокі роздільні згортки, які дозволяють розділити процес згортки на два етапи. Перший етап виконує згортку на кожному каналі окремо, а другий етап з’єднує результати. Це значно знижує обчислювальні витрати [38].

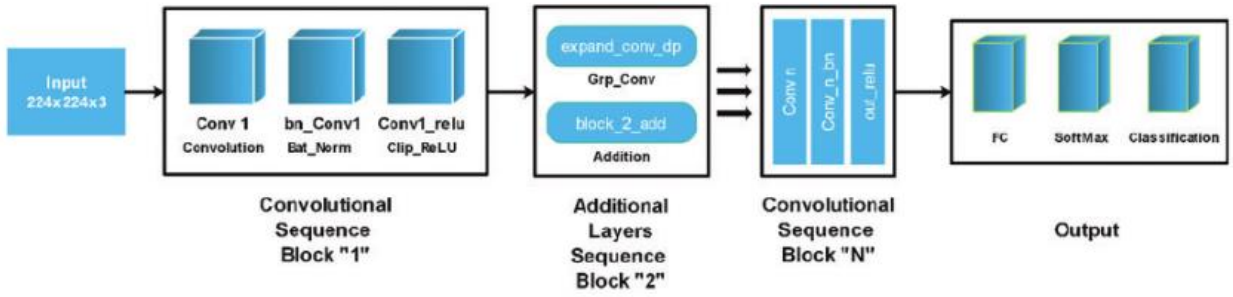

Рисунок 2.11 – Архітектура MobileNetV2 [38]

Крім того, модель MobileNetV2 використовує inverted residual blocks, які включають три основні етапи. Спочатку розмірність вхідних даних зменшується за допомогою згортки 1×1, далі застосовується глибока згортка 3×3, а потім знову відновлюється розмірність за допомогою ще однієї згортки 1×1 [38]. Використання активаційної функції ReLU6 допомагає уникнути проблеми затухання градієнтів. Хоча MobileNetV2 має свої переваги, але вона може показувати нижчу точність на складних завданнях.

Модель EfficientNetB0 (рис. 2.12) є частиною сімейства EfficientNet, яке було створено для вдосконалення архітектур шляхом компаундного масштабування. Цей підхід передбачає одночасне врахування трьох ключових аспектів: глибини, ширини та розміру пікселя з метою досягнення високої

ефективності при меншій кількості параметрів в порівнянні з попередніми версіями. EfficientNetB0 демонструє вражаючі результати на загальноприйнятих наборах даних завдяки своїй здатності раціонально використовувати ресурси [39].

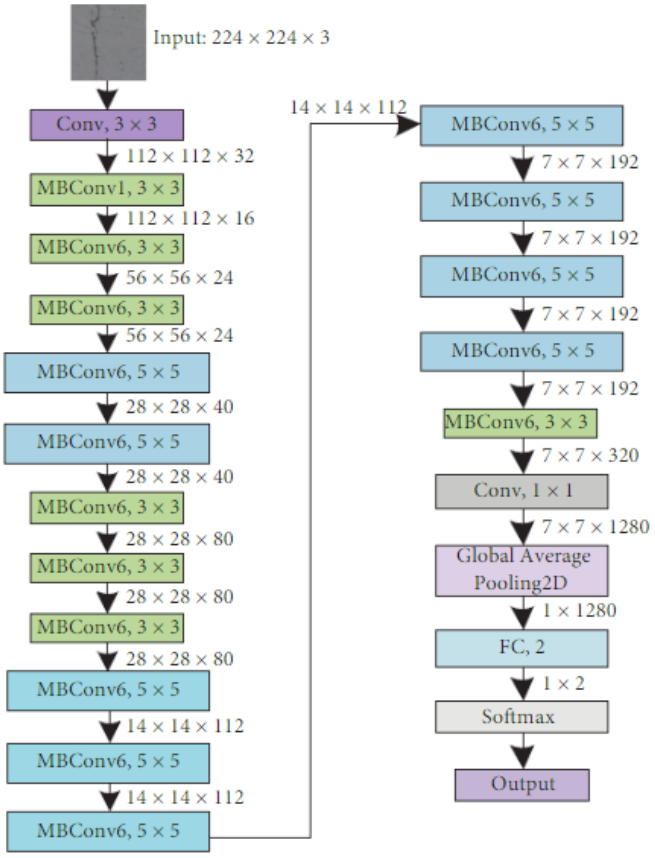

Рисунок 2.12 – Архітектура EfficientNetB0 [39]

Основними перевагами моделі EfficientNetB0 є її висока точність при нижчій обчислювальній складності. Вона також добре масштабується під різні задачі, завдяки чому її можна ефективно використовувати для застосувань у класифікації зображень, розпізнаванні та сегментації об’єктів. Однак обчислювальні вимоги на EfficientNetB0 можуть бути вищими, ніж у MobileNetV2, що може обмежити можливість її використання на пристроях із обмеженими ресурсами [39].

Модель DenseNet121 (рис. 2.13) відрізняється від інших архітектур завдяки щільним з’єднанням між шарами. У цій моделі кожен шар отримує

дані від усіх попередніх шарів, що дозволяє зберігати більше інформації та градієнтів. Така структура значно покращує процес навчання, оскільки дозволяє інформації легко передаватися по всій мережі без втрат [40].

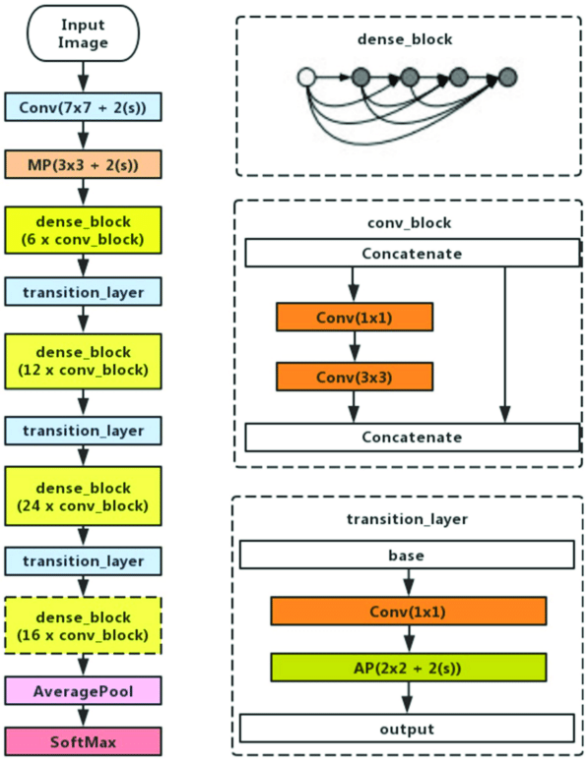

Рисунок 2.13 – Архітектура DenseNet121 [40]

Переваги моделі DenseNet121 включають високу точність та меншу кількість параметрів в порівнянні з традиційними архітектурами з аналогічною глибиною. Це робить DenseNet121 ефективною для завдань класифікації зображень. Однак, з огляду на щільність з'єднань, навчання моделі може вимагати більше пам'яті та часу, особливо на великих наборах даних, що може бути проблемою для пристроїв з обмеженими ресурсами [40].

Модель ResNet50V2 (рис. 2.14) є покращеною версією архітектури ResNet, яка застосовує зворотні зв'язки для запобігання проблемам із затуханням градієнтів. Це досягається шляхом використання ідентичних зав'язків, які дозволяють інформації проходити через різні шари, зберігаючи ефективність роботи у глибоких мережах [41]. Архітектура ResNet50V2 має 50 шарів та показує високу точність на багатьох відомих наборах даних.

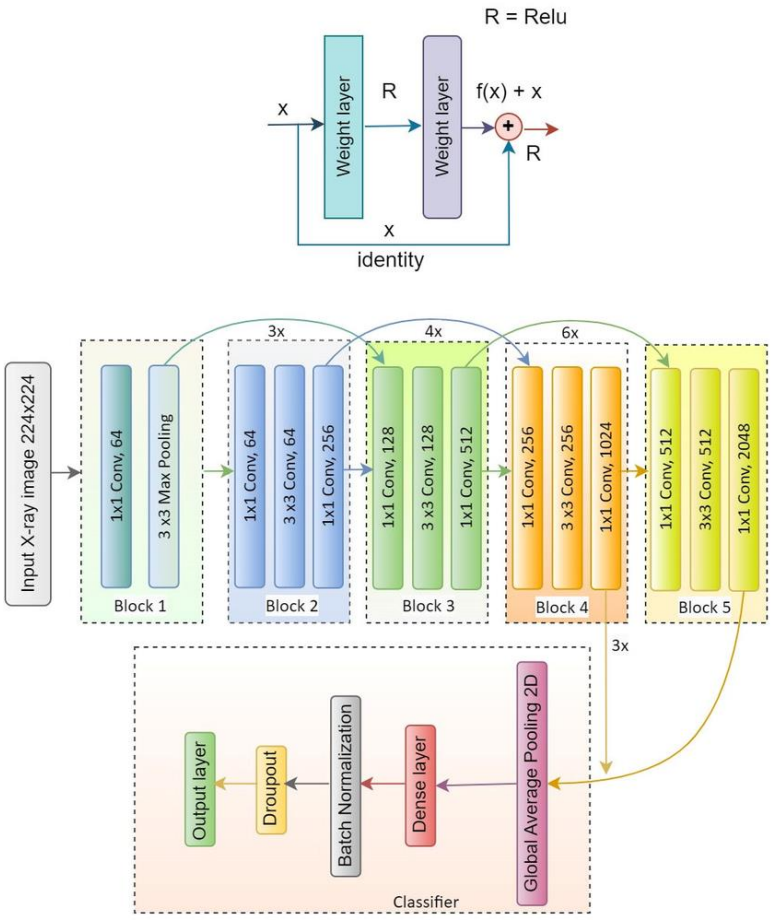

Рисунок 2.14 – Архітектура ResNet50V2 [41]

Серед переваг моделі ResNet50V2 варто відзначити її здатність ефективно навчатися на великих обсягах даних і стабільну продуктивність. Гнучкість цієї моделі дозволяє легко адаптувати її для різних завдань. Проте через велику кількість параметрів та обчислювальну складність, ResNet50V2 може бути менш ефективною на мобільних пристроях, що може призвести до сповільнення інференсу [41].

Модель NASNetMobile (рис. 2.15) оптимізована для мобільних пристроїв і була розроблена за допомогою нейронного архітектурного пошуку. Цей підхід дозволяє автоматично знаходити ефективні конфігурації мережі, що в свою чергу, призводить до покращення продуктивності при зменшенні обчислювальної складності [42].

Основними перевагами NASNetMobile є її адаптивність та здатність досягати високих результатів при значно меншій кількості параметрів, що

робить її особливо придатною для мобільних і вбудованих систем. Однак недостатком є те, що процес пошуку архітектури може вимагати значних обчислювальних ресурсів на етапі навчання, що може бути неприйнятним для деяких мобільних пристроїв [42].

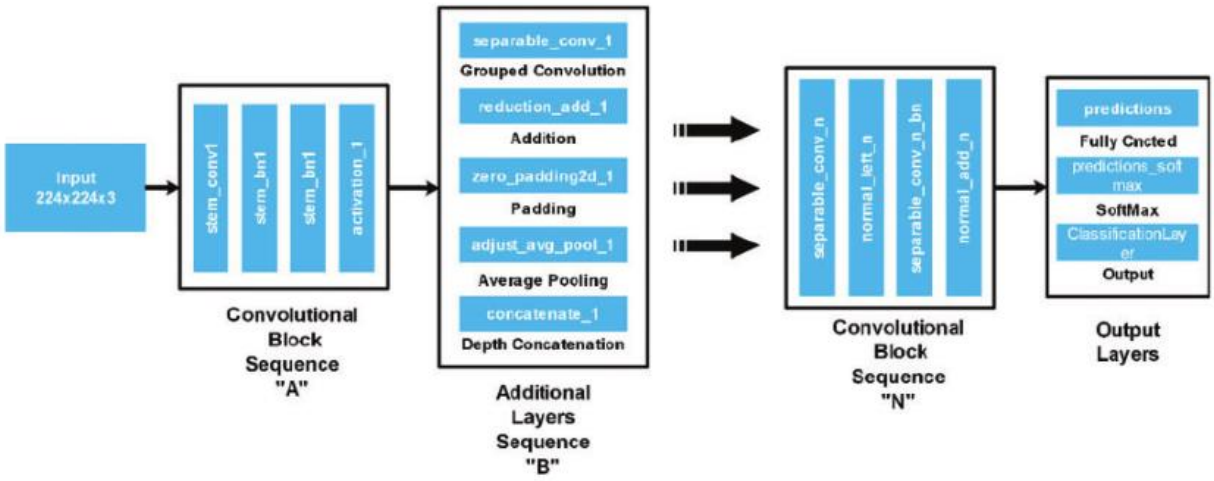

Рисунок 2.15 – Архітектура NASNetMobile [42]

Отже, використання архітектури MobileNetV2, EfficientNetB0, DenseNet121, ResNet50V2 та NASNetMobile залежить від конкретних вимог до точності, швидкості та обчислювальних ресурсів. Кожна з цих моделей має свої унікальні особливості, переваги та недоліки, що робить їх придатними для різних завдань у галузі комп’ютерного зору.

2.4 Опис подальших етапів та оцінка результативності класифікації зображень

В даній кваліфікаційній роботі буде реалізовано класифікацію зображень одягу за допомогою п’яти популярних архітектур CNN: MobileNetV2, EfficientNetB0, DenseNet121, NASNetMobile та ResNet50V2 та оцінювання ефективності класифікації одягу.

Процес класифікації (рис. 2.16) буде включати такі етапи:

Етап 1. Вибір датасету для тренування та тестування. Для даного дослідження використовується власноруч розроблений датасет, що складається із зображень різних категорій одягу. Крім того, можуть бути використані загальнодоступні датасети, які підходять для класифікації зображень одягу.

Етап 2. Підготовка та аугментація даних. Зображення розподіляються на три набори: навчальний, валідаційний та тестовий. Після цього проводиться попередня обробка даних, яка включає масштабування. Також застосовуються різні методи аугментації, такі як обертання, зміна яскравості та дзеркальне відображення.

Етап 3. Налаштування архітектури нейронної мережі. Обирається архітектура, наприклад, MobileNetV2. Модель використовується з попередньо натренованими вагами для реалізації Transfer Learning, що означає ініціалізацію з уже навченою вагою та заморожування певної кількості шарів, щоб запобігти їхньому оновленню під час тренування. Після цього на основі базової моделі будується власна модель, додаючи нові шари для виконання класифікації на основі специфічних для завдання даних. Далі визначаються гіперпараметри навчання, такі як розмір батча, кількість епох, початкова швидкість навчання та оптимізатор Adam.

Етап 4. Навчання нейронної мережі. Модель навчається на тренувальному наборі даних, використовуючи оптимізатор Adam для мінімізації функції втрат, яка визначає різницю між прогнозами та правильними відповідями. Під час тренування модель отримує навчальні дані, обчислює втрати і коригує свої параметри за допомогою алгоритму оптимізації, такого як Adam. Процес повторюється протягом кількох епох, доки модель не покаже стабільні результати на валідаційній вибірці. Використовуються callback-функції для ранньої зупинки тренування, якщо точність на валідаційних даних перестає покращуватись.

Етап 5. Тестування та оцінка моделі. Модель проходить тестування на тестовій вибірці, яка не використовувалася під час тренування. Моделі

оцінюються за допомогою графіків втрат і точності навчання, матриці плутанини та метрик точності, таких як accuracy, precision, recall, F1-score, ROC-AUC та Log Loss.

Етап 6. Квантування моделі для мобільних пристроїв. Зменшити розмір моделі за допомогою квантування, щоб зробити її придатною для мобільних пристроїв.

Етап 7. Повторення Етапів 2–6 на інших архітектурах нейронних мереж.

Етап 8. Порівняння результативності різних архітектур на вибраному наборі даних.

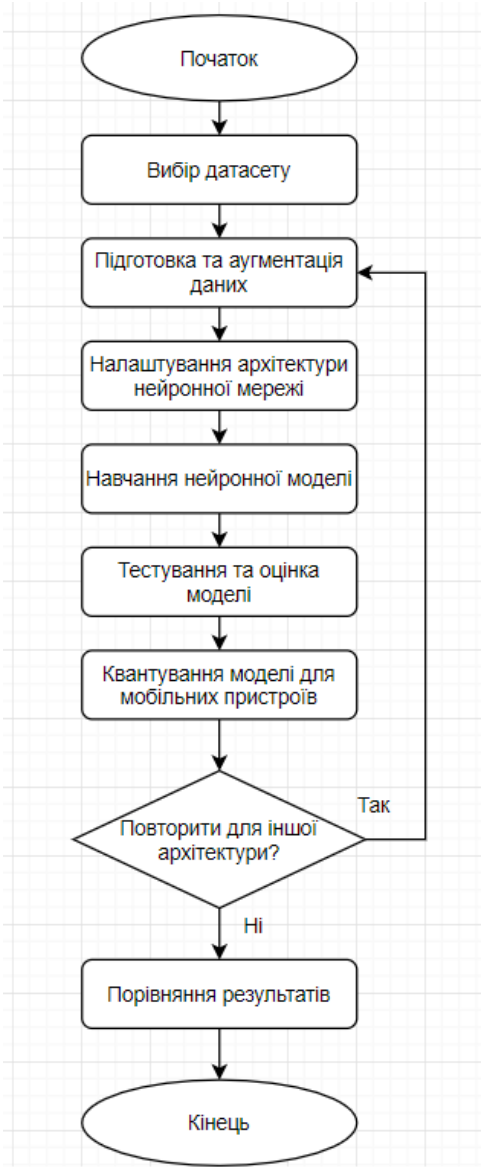

Рисунок 2.16 – Алгоритм процесу класифікації зображень

Матриця плутанини – це квадратна таблиця, яка використовується для оцінки ефективності класифікаційних моделей, відображаючи співвідношення між фактичними та прогнозованими класами в наборі даних [43].

Для класифікації зображень матриця плутанини матиме вигляд, як показано на рисунку 2.17, де:

– P – кількість правильно передбачених зображень;

– TN – кількість правильно не передбачених зображень;

– FP – кількість зображень, які модель передбачила неправильно;

– FN – кількість зображень, які модель пропустила.

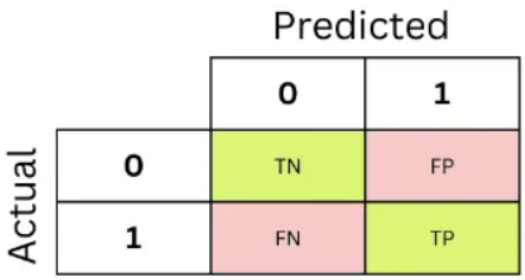

Рисунок 2.17 – Матриця плутанини

Accuracy – метрика, яка показує, наскільки добре модель класифікує дані, обчислюючи частку правильних прогнозів серед загальної кількості спроб. Метрика обчислюється за формулою:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (2.4)$$

Accuracy допомагає оцінити загальну ефективність моделі, але може бути недостатньо інформативною, якщо класи в даних нерівномірно представлені. Наприклад, коли один клас переважає, висока точність може вказувати на те, що модель добре справляється з домінуючим класом, але не обов’язково точно передбачає рідкісні випадки [44]. У таких ситуаціях краще використовувати додаткові метрики, як-от precision, recall, F1-score, ROC-AUC та Log Loss, щоб отримати більш повну картину про ефективність моделі.

Precision – це також метрика, яка показує, наскільки точні позитивні прогнози моделі. Вона визначає, яка частка з усіх прогнозованих позитивних результатів виявилася правильними [44]. Тобто, precision показує, як часто модель помиляється, коли прогнозує позитивний результат. Високий показник precision означає, що модель робить менше помилок, прогнозуючи позитивні результати, однак ця метрика не враховує, скільки позитивних випадків було пропущено. Метрика обчислюється за формулою:

$$Precision = \frac{TP}{TP+FP}. \quad (2.5)$$

Recall – метрика, яка показує, скільки з усіх реальних позитивних випадків модель змогла правильно передбачити. Ця метрика важлива у тих випадках, коли пропуск позитивного результату може мати серйозні наслідки. Високе значення recall означає, що модель не пропускає багато позитивних випадків, але це не завжди говорить про точність прогнозів, оскільки можуть бути помилкові позитивні передбачення [44]. Метрика обчислюється за формулою:

$$Recall = \frac{TP}{TP+FN}. \quad (2.6)$$

F1-score – ще одна метрика, яка об’єднує precision та recall в одне значення, щоб надати збалансовану оцінку продуктивності моделі, коли класифікація є незбалансованою [44]. F1-score є гармонічним середнім між precision і recall, що робить його корисним, коли важливо врахувати як точність позитивних передбачень, так і здатність моделі виявляти всі позитивні випадки. Тобто, F1-score надає кращу оцінку, коли є дисбаланс між precision і recall, адже середнє значення стає меншим, якщо одна з метрик значно нижча за іншу. Метрика обчислюється за формулою:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recal} = 2 \times \frac{TP}{2 \times TP + FP + FN}. \quad (2.7)$$

ROC-AUC – це також метрика, яка використовується для оцінки продуктивності класифікаційних моделей. Вона об’єднує криву характеристик приймача (ROC-криву) та площу під цією кривою (AUC).

ROC-крива відображає графік, який допомагає оцінити, наскільки добре модель відокремлює позитивні та негативні класи в задачах класифікації [45]. Ця крива будується на основі порівняння частки правильно передбачених позитивних прикладів з усіх фактичних позитивних, які обчислюються за формулою:

$$TPR = \frac{TP}{TP + FN} \quad (2.8)$$

та неправильно класифікованих негативних прикладів як позитивні, які обчислюються за формулою:

$$FPR = \frac{FP}{FP + TN}. \quad (2.9)$$

ROC-крива показує, як змінюються ці два показники при різних порогах рішення моделі. Якщо поріг низький, модель прогнозує більше позитивних випадків, що збільшує і TPR, і FPR. Якщо поріг високий, модель стає більш обережною, що зменшує обидва показники.

Ідеальна модель матиме ROC-криву, яка проходить максимально близько до лівого верхнього кута графіка, де TPR високий, а FPR низький. Чим ближче крива до діагоналі (лінії випадкових здогадок), тим менш ефективною є модель.

AUC – метрика, яка показує площу під ROC-кривою. Вона використовується для оцінки якості класифікаційної моделі, яка вимірює, наскільки добре модель відрізняє позитивні приклади від негативних на різних

порогах рішення. Значення AUC можуть коливатися в діапазоні від 0 до 1, де 1 означає, що модель ідеальна та правильно розрізняє позитивні і негативні класи, а 0 – модель більше плутає класи та працює гірше за випадкові здогадки.

Отже, ROC-AUC дозволяє оцінити здатність моделі розрізняти класи на різних порогах рішення. Модель, яка має високий AUC є більш потужною для класифікації, бо вона добре відокремлює позитивні приклади від негативних.

Log Loss – це логарифмічна втрата, метрика, яка вимірює якість класифікаційної моделі, оцінюючи, наскільки добре модель передбачає ймовірності належності об’єкта до певного класу. Вона враховує не тільки те, чи був правильний клас передбачений, але і наскільки близькими були передбачені ймовірності до фактичних результатів [46]. Log Loss обчислюється за формулою:

$$Log\ Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)], \qquad (2.10)$$

де N – загальна кількість прикладів;

y_i – фактичний клас (0 або 1);

p_i – прогнозована ймовірність належності до класу 1.

Якщо модель впевнена в своєму правильному прогнозі і ймовірність близька до 1 для вірного класу, то значення Log Loss буде низьким. Але, якщо модель помиляється і вона впевнена у своєму прогнозі, то Log Loss буде високим.

Отже, чим нижчий показник, тим краща модель. Ця метрика корисна, коли модель видає ймовірності, бо вона враховує не тільки те, що був прогноз правильним, але й наскільки модель була впевнена у своїх прогнозах.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕТОДУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ

3.1 Вибір програмних засобів та інструментів

У рамках кваліфікаційної роботи було розроблено класифікацію одягу за допомогою п’яти популярних архітектур: MobileNetV2, EfficientNetB0, DenseNet121, NASNetMobile та ResNet50V2, а також виконано оцінку їхньої ефективності для класифікації одягу.

Для вирішення завдання було обрано мову програмування Python, яка завдяки своїй гнучкості, простоті та широкій підтримці стала одним із найпопулярніших інструментів у галузі науки про дані, машинного навчання та веброзробки [47]. Завдяки лаконічному синтаксису, Python дозволяє розробникам писати чистий, зрозумілий код, що значно спрощує процес розробки, тестування та підтримки програм.

Python став основою для роботи з нейронними мережами завдяки популярним фреймворкам, таким як TensorFlow, PyTorch, Keras та MXNet. Ці інструменти значно спрощують процес створення, тренування і налаштування складних моделей глибокого навчання. Зокрема, TensorFlow надає широкий спектр можливостей для оптимізації моделей, а також конвертації їх у різні формати, як-от TFLite, що використовується на мобільних пристроях [47].

Крім того, мова Python активно використовується для обробки зображень завдяки таким бібліотекам, як OpenCV, Pillow та imgaug. Бібліотека imgaug дозволяє автоматично генерувати додаткові варіанти зображень за допомогою різних перетворень, збільшуючи обсяг тренувального набору без необхідності в додаткових знімках [47]. Imgaug пропонує великий набір трансформацій, таких як обертання, масштабування, зсув, зміна яскравості та контрасту, а також додавання шумів.

Розробка моделі здійснювалася в середовищі Jupyter Notebook (рис. 3.1). Однією з ключових особливостей Jupyter Notebook є підтримка текстових блоків із розміткою Markdown, завдяки яким можна додавати пояснення, графіки, формули та інші текстові коментарі прямо в робочому зошиті. Це робить його зручним інструментом для створення детальних звітів, які поєднують код та опис методології [48].

Також, Jupyter Notebook підтримує вбудовану візуалізацію даних через бібліотеки, такі як Matplotlib та Seaborn [48]. Це дозволяє створювати графіки та діаграми, що допомагає швидко оцінювати ефективність моделі, аналізувати проміжні результати або переглядати властивості набору даних.

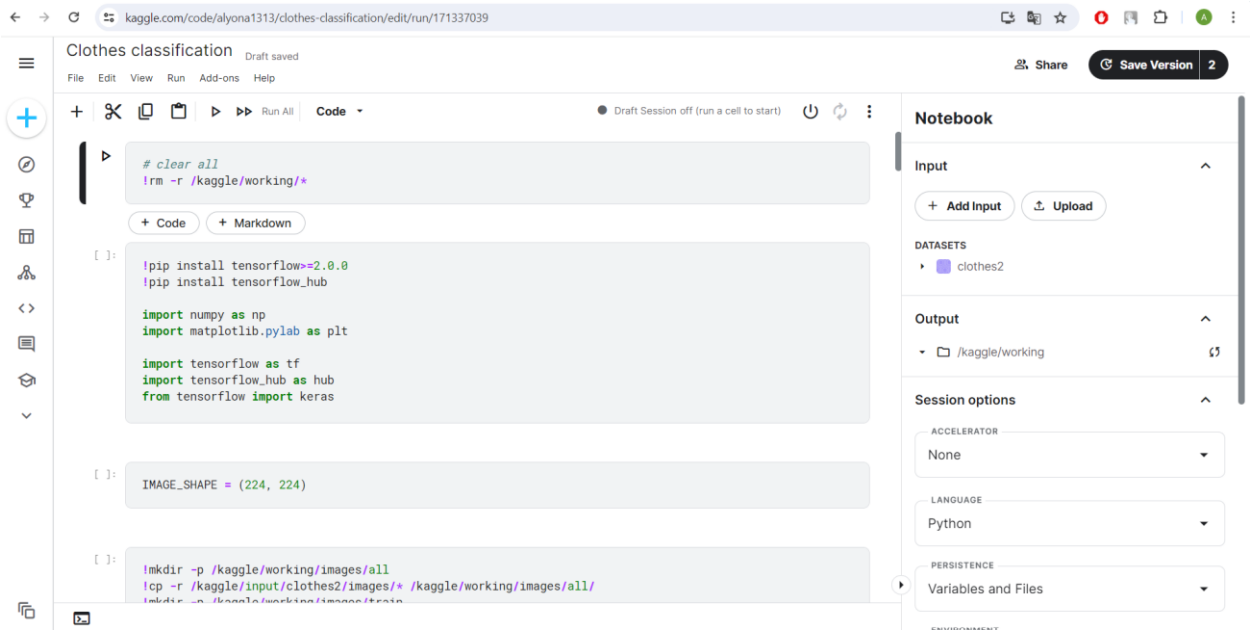

Рисунок 3.1 – Приклад використання Jupyter Notebook на платформі Kaggle

Для обробки великих обсягів даних та прискорення навчання моделей було використано платформу Kaggle. Вона надає можливість працювати з графічними процесорами, що значно скорочує час, необхідний для тренування моделей [49]. Крім того, на Kaggle доступні готові набори даних, які можна використовувати для досліджень, аналізу та навчання моделей. Платформа також містить інтерактивний блокнот, що дозволяє запускати код

безпосередньо на сервері, не потребуючи завантаження даних на локальний комп’ютер.

Таким чином, для виконання завдання було обрано мову програмування Python із застосуванням бібліотек TensorFlow та imgaug. Розробка здійснювалася в інтерактивному середовищі Jupyter Notebook, а зберігання та обробка зображень виконувалися на платформі Kaggle, яка забезпечила доступ до графічних процесорів (GPU) для прискорення процесу навчання моделей.

3.2 Створення набору даних одягу

У цьому дослідженні для класифікації одягу було прийнято рішення створити власний набір даних, оскільки наявні готові набори не відповідали специфічним вимогам, необхідним для якісного навчання моделі. Основною причиною стала недостатня різноманітність і нерівномірний розподіл зображень за категоріями, що є критично важливим для забезпечення збалансованого навчання та оптимальної роботи моделі.

Для створення якісного набору даних було обрано 12 основних типів одягу, що охоплюють різні категорії: верхній одяг (футболки, світшоти, худі, светри, жилети), нижній одяг (штани, спідниці, шорти) та взуття (черевики, чоботи, кросівки). Таке структурування категорій дозволяє моделі враховувати різноманітність стилів, сезонності та призначення одягу (рис. 3.2–3.5).

Особливу увагу приділено рівномірному розподілу зображень між класами, що забезпечує стабільність навчання моделі, знижуючи ризик перенавчання на окремих класах і водночас недостатнього навчання на інших. Для кожної категорії було відібрано зображення з однаковими умовами зйомки та роздільною здатністю, щоб забезпечити одноманітність та високу якість зображень.

Рисунок 3.2 – Приклад датасету одягу: верхній одяг

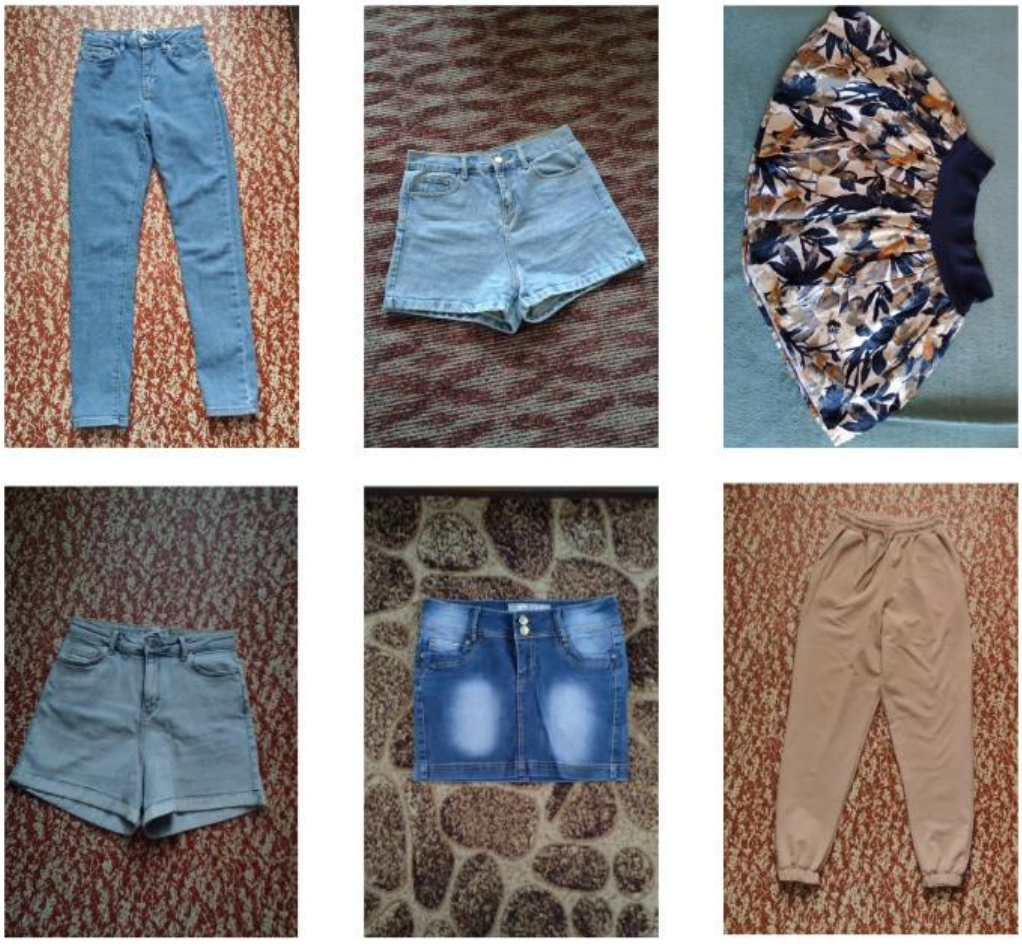

Рисунок 3.3 – Приклад датасету одягу: нижній одяг

Рисунок 3.4 – Приклад датасету одягу: взуття

Для зручності зберігання та легкого доступу до датасету було обрано хмарне сховище Kaggle. Це дозволяє зберігати великі обсяги даних та легко інтегрувати їх у процес навчання моделей, забезпечуючи безперебійний доступ до зображень на етапах тренування, тестування та валідації. Структура зберігання на Kaggle проілюстрована на рисунку 3.5.

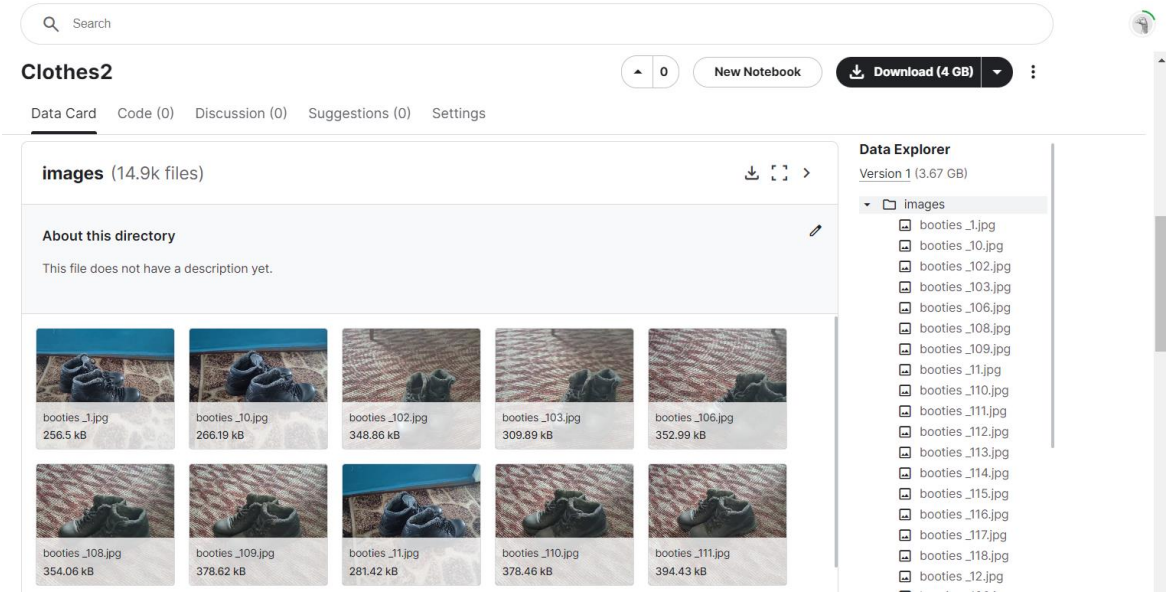

Рисунок 3.5 – Хмарне сховище для цифрової колекції одягу

3.3 Програмна реалізація класифікації одягу

Процес класифікації одягу розпочався з розподілу власного датасету на навчальний, валідаційний та тестовий набори, що дозволило забезпечити об’єктивну оцінку моделей на невідомих даних (рис. 3.6).

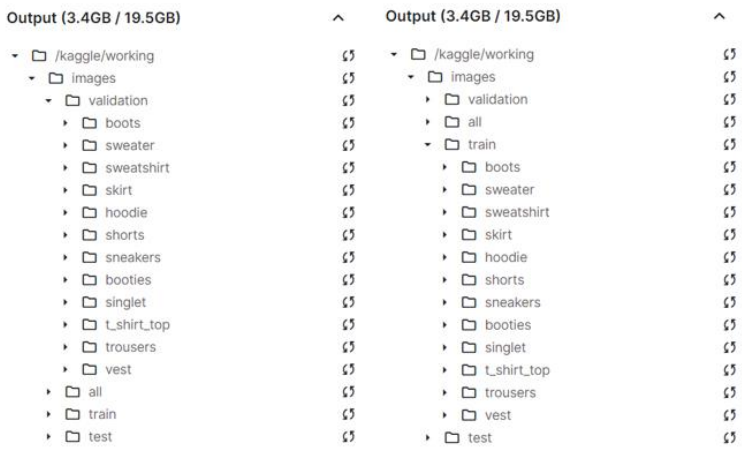

Рисунок 3.6 – Ієрархія датасету для класифікації одягу

Після цього зображення пройшли попередню обробку, включаючи аугментацію (рис. 3.7) для збільшення обсягу даних, а також масштабування (рис. 3.8).

```python
from imgaug import augmenters as iaa
import cv2

aug = iaa.Sequential(
    [
        iaa.Fliplr(0.5),
        iaa.Crop(percent=(0, 0.1)),
        iaa.Affine(scale=(0.8, 1.2), rotate=(-25, 25)),
        iaa.Multiply((0.8, 1.2)),
        iaa.LinearContrast((0.75, 1.5))
    ],
)

def augment_images(directory):
    for category in categories:
        path = os.path.join(directory, category)
        for filename in os.listdir(path):
            img = cv2.imread(os.path.join(path, filename))
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            augmented_img = aug.augment_image(img)
            new_filename = os.path.splitext(filename)[0] + '_augmented' + os.path.splitext(filename)[1]
            cv2.imwrite(os.path.join(path, new_filename), augmented_img)

augment_images(prepared_dir)
```

Рисунок 3.7 – Лістинг коду аугментації зображень

```python
def rescale_image(image, label):
    image = tf.cast(image, tf.float32) / 255.0
    return image, label

if model_name == 'EfficientNetB0':
    pass
elif model_name == 'NASNetMobile':
    train_dataset = train_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    val_dataset = val_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    test_dataset = test_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
elif model_name == 'DenseNet121':
    train_dataset = train_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    val_dataset = val_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    test_dataset = test_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
elif model_name == 'ResNet50V2':
    train_dataset = train_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    val_dataset = val_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    test_dataset = test_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
elif model_name == 'MobileNetV2':
    train_dataset = train_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    val_dataset = val_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
    test_dataset = test_dataset.map(rescale_image, num_parallel_calls=tf.data.AUTOTUNE)
```

Рисунок 3.8 – Лістинг коду масштабування зображень

Кожна з обраних архітектур CNN використовувала Transfer Learning з попередньо натренованими вагами, що скоротило час навчання та покращило результати класифікації (рис. 3.9).

```python
if model_name == 'EfficientNetB0':
    from tensorflow.keras.applications import EfficientNetB0
    model = EfficientNetB0(include_top=False, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

elif model_name == 'NASNetMobile':
    from tensorflow.keras.applications import NASNetMobile
    model = NASNetMobile( include_top=False, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

elif model_name == 'DenseNet121':
    from tensorflow.keras.applications import DenseNet121
    model = DenseNet121(include_top=False, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

elif model_name == 'ResNet50V2':
    from tensorflow.keras.applications import ResNet50V2
    model = ResNet50V2(include_top=False, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))

elif model_name == 'MobileNetV2':
    from tensorflow.keras.applications import MobileNetV2
    model = MobileNetV2(include_top=False, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3))
```

Рисунок 3.9 – Лістинг коду вибору попередньо навчених моделей для Transfer Learning

Базова модель була доповнена новими шарами, такими як GlobalAveragePooling2D для зменшення кількості параметрів, двома щільними шарами (Dense) з активацією ReLU для навчання нових ознак, шарами Dropout

для зменшення перенавчання та вихідним шаром Dense з активацією softmax, що відповідає за класифікацію на задану кількість класів (рис. 3.10).

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *

my_model = Sequential([
        model,
        GlobalAveragePooling2D(),
        Dense(512, activation='relu'),
        Dropout(0.2),
        Dense(512, activation='relu'),
        Dropout(0.2),
        Dense(num_classes, activation='softmax'),
])

my_model.summary()
```

Рисунок 3.10 – Лістинг коду доповнення базової моделі новими шарами

Також були налаштовані гіперпараметри навчання, такі як розмір батча (рис. 3.11), кількість епох (рис. 3.12), функція втрат, початкова швидкість навчання та оптимізатор Adam (рис. 3.13).

```
import tensorflow as tf

model_name = 'ResNet50V2'

if model_name == 'EfficientNetB0':
    BATCH_SIZE = 4
elif model_name == 'NASNetMobile':
    BATCH_SIZE = 32
elif model_name == 'DenseNet121':
    BATCH_SIZE = 32
elif model_name == 'ResNet50V2':
    BATCH_SIZE = 32
elif model_name == 'MobileNetV2':
    BATCH_SIZE = 32

SEED = 1
IMAGE_SIZE = 224
```

Рисунок 3.11 – Лістинг коду розміру батча

```
if model_name == 'EfficientNetB0':
    EPOCHS = 3
elif model_name == 'NASNetMobile':
    EPOCHS = 7
elif model_name == 'DenseNet121':
    EPOCHS = 4
elif model_name == 'ResNet50V2':
    EPOCHS = 7
elif model_name == 'MobileNetV2':
    EPOCHS = 7
```

Рисунок 3.12 – Лістинг коду кількості епох навчання моделей

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import CategoricalCrossentropy


model_loss = CategoricalCrossentropy(from_logits=False)

my_model.compile(loss=model_loss, optimizer=Adam(learning_rate=0.001),metrics=metrics)
```

Рисунок 3.13 – Лістинг коду вибору функції втрат, оптимізатора та швидкості навчання для навчання моделей

Навчання моделей проводилося на тренувальному наборі даних. Під час навчання використовувалися функції для ранньої зупинки, що дозволило уникнути перенавчання моделей та забезпечити стабільні результати на валідаційній вибірці (рис. 3.14).

```
from tensorflow.keras.callbacks import EarlyStopping


es = EarlyStopping(patience=3, monitor='val_loss')

my_model.fit( train_dataset, epochs=EPOCHS, validation_data = val_dataset, callbacks=[es])
```

Рисунок 3.14 – Лістинг коду навчання моделей

Після завершення навчання кожна модель проходила тестування на тестовій вибірці, де оцінювалася за допомогою графіків втрат і точності навчання (рис. 3.15), матриці плутанини, accuracy, precision, recall, F1-score, ROC-AUC та Log Loss (рис. 3.16). Це дозволило детально проаналізувати

ефективність класифікації та визначити сильні й слабкі сторони кожної архітектури.

```python
def plot_hist(hist):
    plt.figure(figsize=(12, 16))

    plt.subplot(4, 2, 2)
    plt.plot(hist.history['accuracy'], label='Train')
    plt.plot(hist.history['val_accuracy'], label='Validation')
    plt.title('Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend()

    plt.subplot(4, 2, 1)
    plt.plot(history.history['loss'], label='Train')
    plt.plot(history.history['val_loss'], label='Validation')
    plt.title('Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend()

plot_hist(history)
```

Рисунок 3.15 – Лістинг коду графіків втрат і точності навчання

```python
def evaluate_model(y_true, y_pred, y_proba, model_description):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred,
                                average='weighted', zero_division=0)
    recall = recall_score(y_true, y_pred,
                          average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred,
                  average='weighted', zero_division=0)
    roc_auc = roc_auc_score(y_true, y_proba,
                            average='weighted', multi_class='ovr')
    logloss = log_loss(y_true, y_proba)

    print(f"--- Metrics for {model_description} ---")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}")
    print(f"Log Loss: {logloss:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred,
                                target_names=class_names, zero_division=0))

    # Confusion Matrix
    conf_matrix = confusion_matrix(y_true, y_pred, normalize='true')
    plt.figure(figsize=(10, 8))
    sns.heatmap(conf_matrix, annot=True, fmt='.2f', cmap='Blues',
                xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title(f'Confusion Matrix for {model_description}')
    plt.show()
```

Рисунок 3.16 – Лістинг коду для оцінки моделі з використанням метрик і матриці плутанини

Для адаптації моделей до мобільних пристроїв було проведено квантування (рис. 3.17), що дозволило зменшити їх розмір без суттєвої втрати точності. Цей етап є важливим для інтеграції систем автоматичної класифікації одягу в мобільні застосунки, забезпечуючи їхню ефективність та швидкодію на обмежених апаратних ресурсах.

```
def representative_data_gen():
    for idx, (input_value, _) in enumerate(train_dataset.take(300)):
        max_value = tf.reduce_max(input_value).numpy()
        min_value = tf.reduce_min(input_value).numpy()

        if model_name == 'EfficientNetB0':
            input_value = tf.cast(input_value, tf.float32)
        else:
            input_value = tf.cast(input_value, tf.float32)
        yield [input_value]

converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_data_gen

converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.uint8
converter.inference_output_type = tf.uint8

quant_tflite_model = converter.convert()

quant_tflite_model_path = os.path.join(results_dir, 'model_quant.tflite')
with open(quant_tflite_model_path, 'wb') as f:
    f.write(quant_tflite_model)
print(f"Quantized TFLite model saved at: {quant_tflite_model_path}")
```

Рисунок 3.17 – Лістинг коду для квантування моделей

Після проведення всіх етапів комп’ютерного моделювання було здійснено порівняння ефективності різних архітектур CNN.

3.4 Аналіз точності і ефективності класифікації одягу

Аналіз класифікації одягу спрямоване на оцінку точності та ефективності моделей, а також на перевірку їхньої здатності до роботи з

реальними даними. Процес тестування буде поділений на кілька етапів, де кожен з етапів забезпечує комплексний аналіз ефективності моделей.

Перший етап забезпечує візуалізацію процесу навчання моделей на графіках, які показують зміну втрат та точності на тренувальному і валідаційному наборах даних для різних архітектур CNN (рис. 3.18–3.22). На графіках втрат видно, як модель поступово зменшує втрати, а на графіках точності – як збільшується точність класифікації.

На графіках втрат та точності для архітектури EfficientNetB0 (рис. 3.18) можна спостерігати стабільне зменшення втрат протягом перших трьох епох. Валідаційні втрати також знижуються, але мають певні коливання на другій епосі, що може свідчити про невеликі складнощі з узагальненням на валідаційних даних. Точність для тренувального набору швидко зростає, досягаючи майже 99%, а валідаційна точність підтримує стабільно високий рівень.

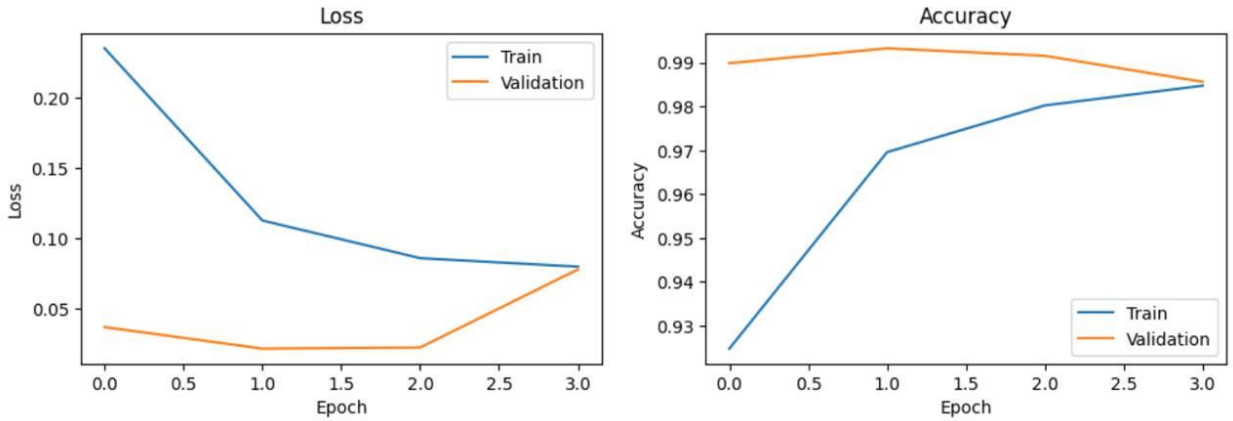

Рисунок 3.18 – Графіки втрат та точності для архітектури EfficientNetB0

На графіках втрат та точності для архітектури NASNetMobile (рис. 3.19) можна спостерігати поступове зниження втрат протягом 7 епох тренування. На тренувальному та валідаційному наборах втрати поступово знижуються до низьких зачень, що вказує на хороше навчання моделі. Також точність поступово збільшується до 97%, що свідчить про ефективну класифікацію.

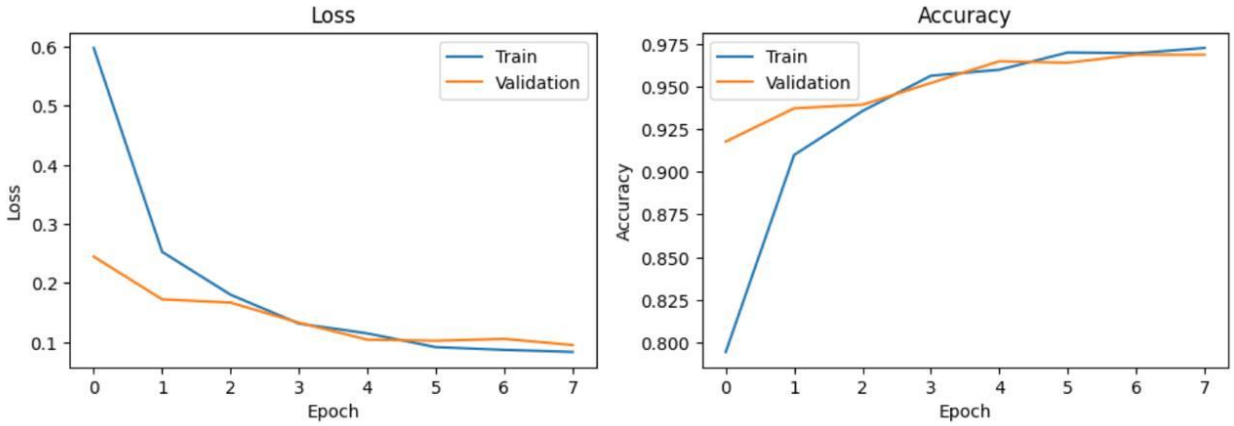

Рисунок 3.19 – Графіки втрат та точності для архітектури NASNetMobile

На графіках втрат та точності для архітектури ResNet50V2 (рис. 3.20) можна спостерігати стабільне зниження втрат на тренувальному наборі, що свідчить про поступове покращення моделі. На валідаційних даних є деякі коливання втрат на етапах 3–5, що свідчать про невелику схильність до перенавчання. Точність на тренувальному наборі досягає майже 98%, а валідаційна точність підтримує стабільно високий рівень.

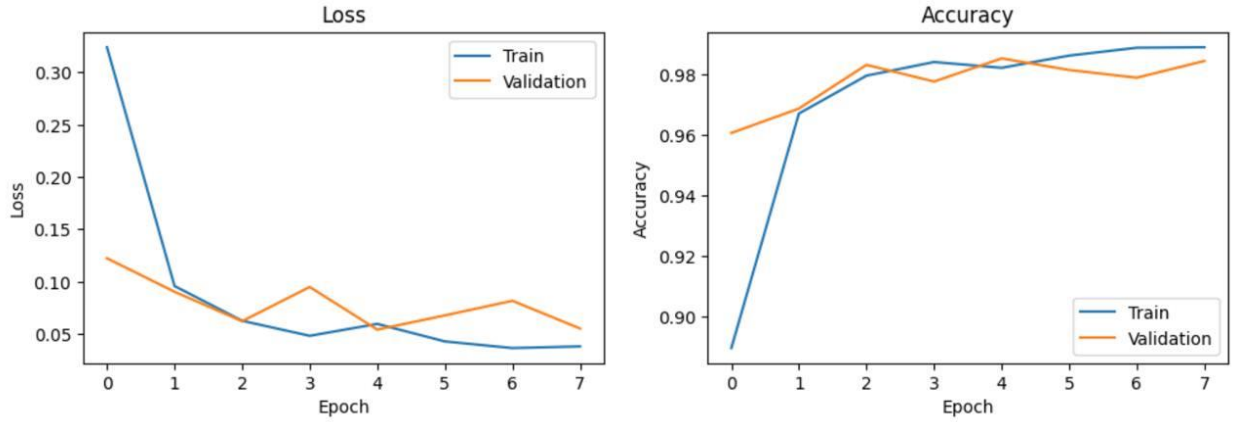

Рисунок 3.20 – Графіки втрат та точності для архітектури ResNet50V2

На графіках втрат та точності для архітектури MobileNetV2 (рис. 3.21) можна спостерігати найбільш стабільне зниження втрат на тренувальному та валідаційному наборах, що вказує на ефективне навчання. Точність на тренувальному наборі зростає до 98%, а на валідаційному наборі – до 97%.

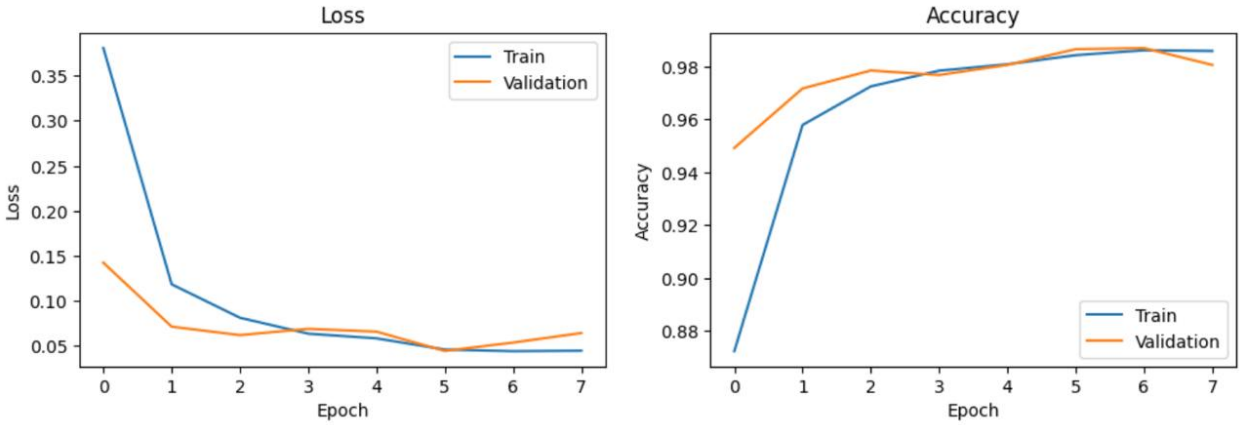

Рисунок 3.21 – Графіки втрат та точності для архітектури MobileNetV2

На графіках втрат та точності для архітектури DenseNet121 (рис. 3.22) можна спостерігати, що модель стабільно зменшує втрати на тренувальному наборі, досягаючи дуже низьких значень після шостої епохи. Втрати на валідаційних даних також знижуються, але мають незначні коливання на останніх епохах, що свідчать про можливі труднощі з узагальненням. Точність тренувального набору зростає до 98%, а точність на валідаційному наборі демонструє схожу динаміку, що вказує на хорошу узгодженість між тренувальними та валідаційними результатами.

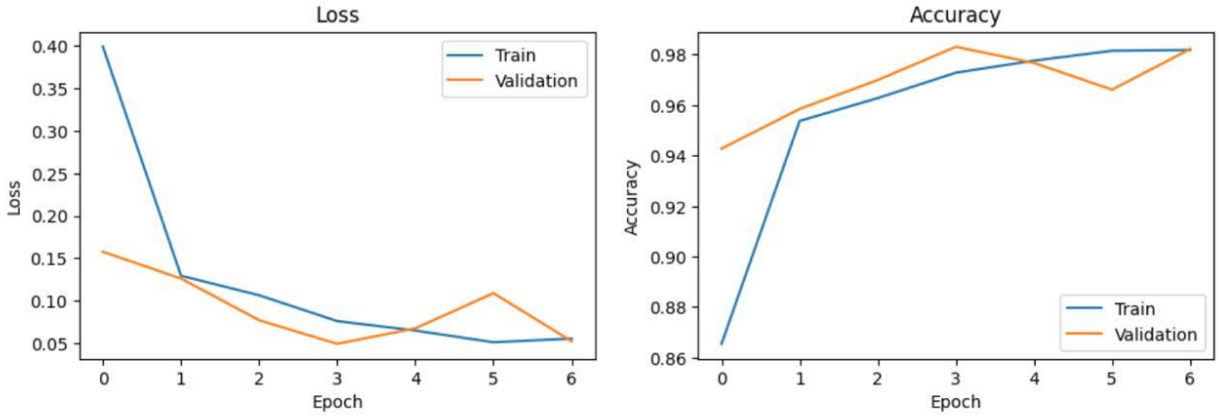

Рисунок 3.22 – Графіки втрат та точності для архітектури DenseNet121

Другий етап тестування полягає у застосуванні архітектур до незалежного тестового набору даних, що складається з 12 категорій одягу.

Тестовий набір був сформований з 10% загальних даних, які не використовувалися під час навчання моделей. Моделі оцінювалися за метриками, таких як accuracy, precision, recall, F1-score, ROC-AUC та Log Loss.

Архітектура EfficientNetB0 (рис. 3.23) продемонструвала високу точність класифікації досягнувши значення 0,99. Середній показник precision склав 0,99, f1-score – 0,99, recall – 0,99. ROC-AUC досяг 0,999, що свідчить про майже ідеальне відокремлення класів. Log Loss становив 0,034, що свідчить про відносно низькі помилки моделі на тестовому наборі.

	precision	recall	f1-score	support
booties	0.96	1.00	0.98	52
boots	1.00	0.94	0.97	105
hoodie	0.99	0.99	0.99	435
shorts	1.00	0.99	0.99	181
singlet	0.99	1.00	1.00	106
skirt	1.00	0.99	0.99	89
sneakers	0.98	1.00	0.99	104
sweater	0.99	0.99	0.99	287
sweatshirt	0.99	0.98	0.99	361
t_shirt_top	0.99	0.99	0.99	277
trousers	0.99	1.00	0.99	287
vest	1.00	0.99	0.99	90
accuracy			0.99	2374
macro avg	0.99	0.99	0.99	2374
weighted avg	0.99	0.99	0.99	2374

```
Accuracy: 0.9899
Precision: 0.9900
Recall: 0.9899
F1 Score: 0.9899
ROC AUC: 0.9999
Log Loss: 0.0338
```

Рисунок 3.23 – Звіт про класифікацію одягу архітектури EfficientNetB0 у форматі SavedModel

Архітектура NASNetMobile (рис. 3.24) продемонструвала високу точність класифікації досягнувши значення 0,967. Середній показник precision склав 0,967, f1-score – 0,973, recall – 0,967. ROC-AUC досяг 0,999, що свідчить

про високу здатність моделі правильно розрізняти класи. Log Loss становив 0,104, що вказує на незначні помилки при передбаченні.

	precision	recall	f1-score	support
booties	0.94	0.96	0.95	52
boots	0.99	0.97	0.98	105
hoodie	0.98	0.96	0.97	435
shorts	0.98	0.92	0.95	181
singlet	0.98	0.97	0.98	106
skirt	0.96	1.00	0.98	89
sneakers	1.00	1.00	1.00	104
sweater	0.93	0.99	0.96	287
sweatshirt	0.97	0.94	0.96	361
t_shirt_top	0.97	0.96	0.97	277
trousers	0.96	1.00	0.98	287
vest	0.98	0.93	0.95	90
accuracy			0.97	2374
macro avg	0.97	0.97	0.97	2374
weighted avg	0.97	0.97	0.97	2374

```
Accuracy: 0.9667228306655434
Precision: 0.9672853480745763
Recall: 0.9667228306655434
F1 Score: 0.9666605281844561
ROC AUC: 0.999186349011707
Log Loss: 0.10428198196128816
```

Рисунок 3.24 – Звіт про класифікацію одягу архітектури NASNetMobile у форматі SavedModel

Архітектура DenseNet121 (рис. 3.25) продемонструвала дуже високу точність класифікації досягнувши значення 0,986. Середній показник precision склав 0,986, f1-score – 0,986, recall – 0,986. ROC-AUC досяг 0,999, що свідчить про високу здатність моделі правильно розрізняти класи. Log Loss становив 0,06, що є найкращим показником серед усіх моделей і вказує на мінімальні помилки при класифікації.

Архітектура ResNet50V2 (рис. 3.26) продемонструвала дуже гарні результати, досягнувши точності 0,99. Середній показник precision склав 0,99, f1-score – 0,99, recall – 0,99. ROC-AUC досяг 0,999, що свідчить про високу здатність моделі правильно розрізняти класи. Log Loss склав 0,061, що свідчить про дуже низькі помилки при передбаченні.

	precision	recall	f1-score	support
booties	0.98	0.96	0.97	52
boots	0.98	0.99	0.99	105
hoodie	1.00	0.98	0.99	435
shorts	0.99	0.98	0.99	181
singlet	1.00	0.99	1.00	106
skirt	0.97	0.99	0.98	89
sneakers	1.00	1.00	1.00	104
sweater	1.00	0.98	0.99	287
sweatshirt	0.98	0.97	0.97	361
t_shirt_top	0.97	1.00	0.98	277
trousers	0.97	1.00	0.99	287
vest	1.00	1.00	1.00	90
accuracy			0.99	2374
macro avg	0.99	0.99	0.99	2374
weighted avg	0.99	0.99	0.99	2374

```
Accuracy: 0.9856781802864364
Precision: 0.9858559294781808
Recall: 0.9856781802864364
F1 Score: 0.9856833486631228
ROC AUC: 0.9996991505781482
Log Loss: 0.06048714276107507
```

Рисунок 3.25 – Звіт про класифікацію одягу архітектури DenseNet121 у форматі SavedModel

	precision	recall	f1-score	support
booties	0.94	0.98	0.96	52
boots	1.00	0.93	0.97	105
hoodie	0.99	1.00	0.99	435
shorts	0.99	0.99	0.99	181
singlet	1.00	0.99	1.00	106
skirt	0.98	0.98	0.98	89
sneakers	0.96	1.00	0.98	104
sweater	0.99	1.00	0.99	287
sweatshirt	0.99	0.99	0.99	361
t_shirt_top	0.99	0.99	0.99	277
trousers	1.00	1.00	1.00	287
vest	1.00	0.99	0.99	90
accuracy			0.99	2374
macro avg	0.99	0.99	0.99	2374
weighted avg	0.99	0.99	0.99	2374

```
Accuracy: 0.9903117101937658
Precision: 0.9904361390322075
Recall: 0.9903117101937658
F1 Score: 0.9902924753462962
ROC AUC: 0.9995252863187506
Log Loss: 0.06065776418996972
```

Рисунок 3.26 – Звіт про класифікацію одягу архітектури ResNet50V2 у форматі SavedModel

Архітектура MobileNetV2 (рис. 3.27) також продемонструвала високі результати, досягнувши точності 0,98. Середній показник precision склав 0,98, f1-score – 0,98, recall – 0,98. ROC-AUC досяг 0,999, що свідчить про високий рівень правильного відокремлення класів. Log Loss склав 0,054 що також є низьким показником помилок при передбаченні.

	precision	recall	f1-score	support
booties	0.96	0.90	0.93	52
boots	1.00	0.97	0.99	105
hoodie	0.99	0.97	0.98	435
shorts	0.98	0.97	0.98	181
singlet	0.98	1.00	0.99	106
skirt	0.97	0.99	0.98	89
sneakers	0.95	1.00	0.98	104
sweater	1.00	0.98	0.99	287
sweatshirt	0.96	0.98	0.97	361
t_shirt_top	0.98	0.97	0.98	277
trousers	1.00	1.00	1.00	287
vest	0.98	1.00	0.99	90
accuracy			0.98	2374
macro avg	0.98	0.98	0.98	2374
weighted avg	0.98	0.98	0.98	2374

```
Accuracy: 0.9802
Precision: 0.9804
Recall: 0.9802
F1 Score: 0.9802
ROC AUC: 0.9998
Log Loss: 0.0541
```

Рисунок 3.27 – Звіт про класифікацію одягу архітектури MobileNetV2 у форматі SavedModel

Третій етап присвячений детальному вивченню помилок моделей за допомогою матриць плутанини. Це дозволить візуалізувати, як моделі класифікують різні категорії одягу та де допускають помилки (рис. 3.28–3.32).

Архітектура EfficientNetB0 (рис. 3.28) продемонструвала високі результати з класифікації більшості категорій, таких як черевики, майки, кросівки та штани, досягнувши точності 100%. Однак виникли певні труднощі при класифікації категорії чоботи, де точність склала лише 94%. Чоботи були переплутані з худі, кросівками, футболками та штанами.

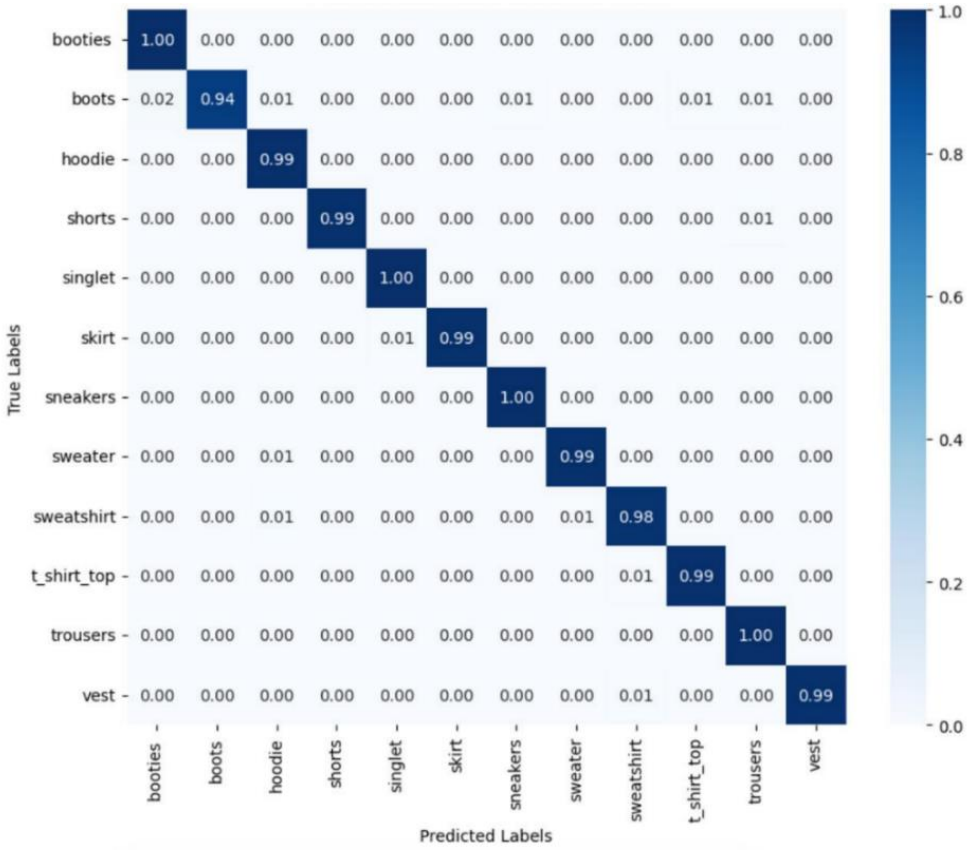

Рисунок 3.28 – Матриця плутанини для архітектури EfficientNetB0 у форматі SavedModel

Архітектура NASNetMobile (рис. 3.29) продемонструвала високі результати в класифікації більшості категорій, таких як спідниці, кросівки та штани, з точністю 100%. Однак, деякі категорії, такі як шорти, жилети та світшоти показали дещо нижчу точність – 92%, 93% та 94% відповідно, що свідчить про певні складнощі з їх класифікацією. Шорти були помилково класифіковані як худі, спідниці, светри, штани та жилети. Світшоти були переплутані з худі, светрами, футболками та штанами. Жилети викликали труднощі при класифікації з худі, шортами та светрами.

Архітектура DenseNet121 (рис. 3.30) продемонструвала дуже високі результати. Точність класифікації для більшості категорій склала від 96% до 100%, що свідчить про високу ефективність моделі. Незначні помилки виникли при класифікації черевиків, які були сплутані з чоботами, що вказує на потребу в додатковому опрацюванні цих категорій.

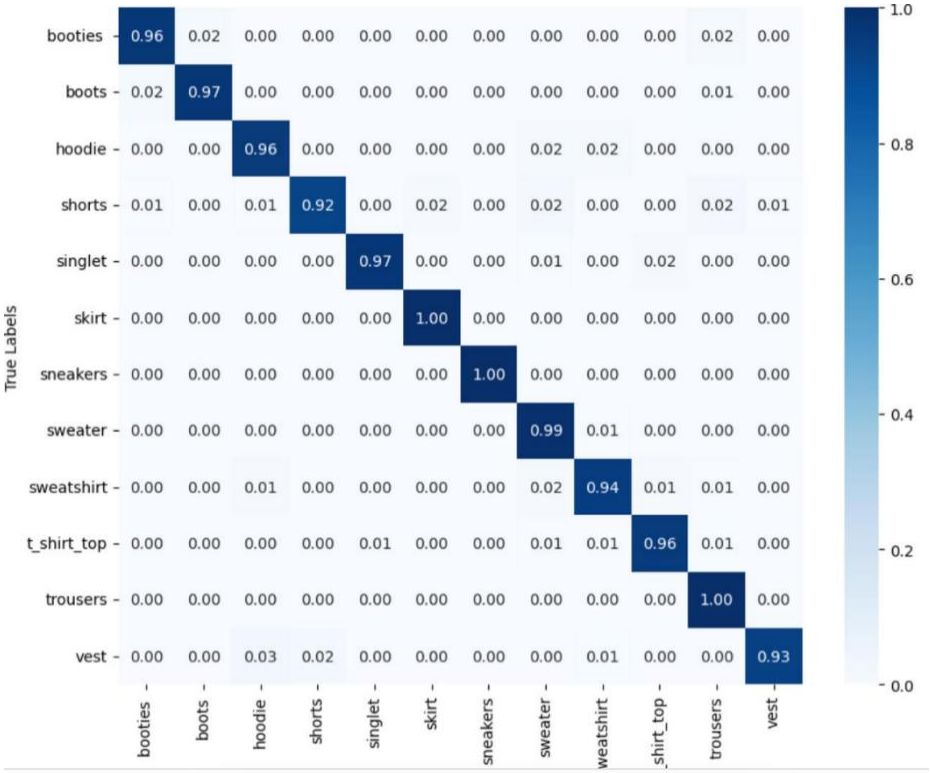

Рисунок 3.29 – Матриця плутанини для архітектури NASNetMobile у форматі SavedModel

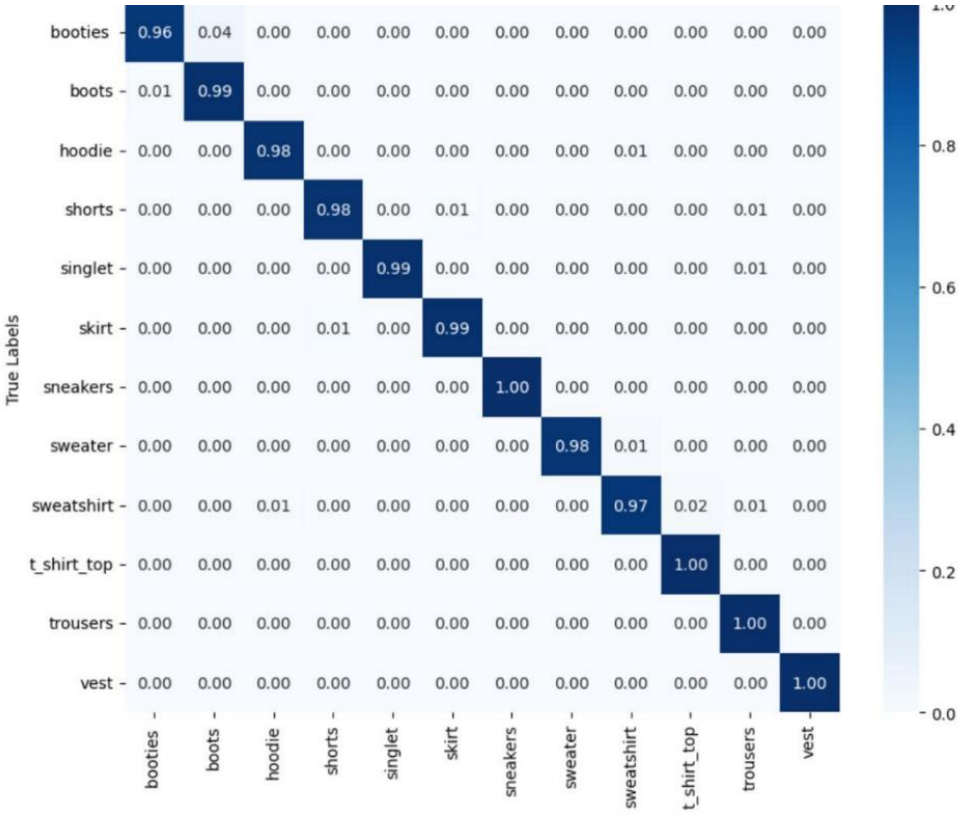

Рисунок 3.30 – Матриця плутанини для архітектури DenseNet121 у форматі SavedModel

Архітектура ResNet50V2 (рис. 3.31) продемонструвала найбільш стабільні результати серед усіх архітектур з точністю класифікації у межах 98%–100% для більшості категорій. Проте виникли незначні помилки у класифікації чобіт, де точність склала 93%. Чоботи були помилково класифіковані як черевики та кросівки.

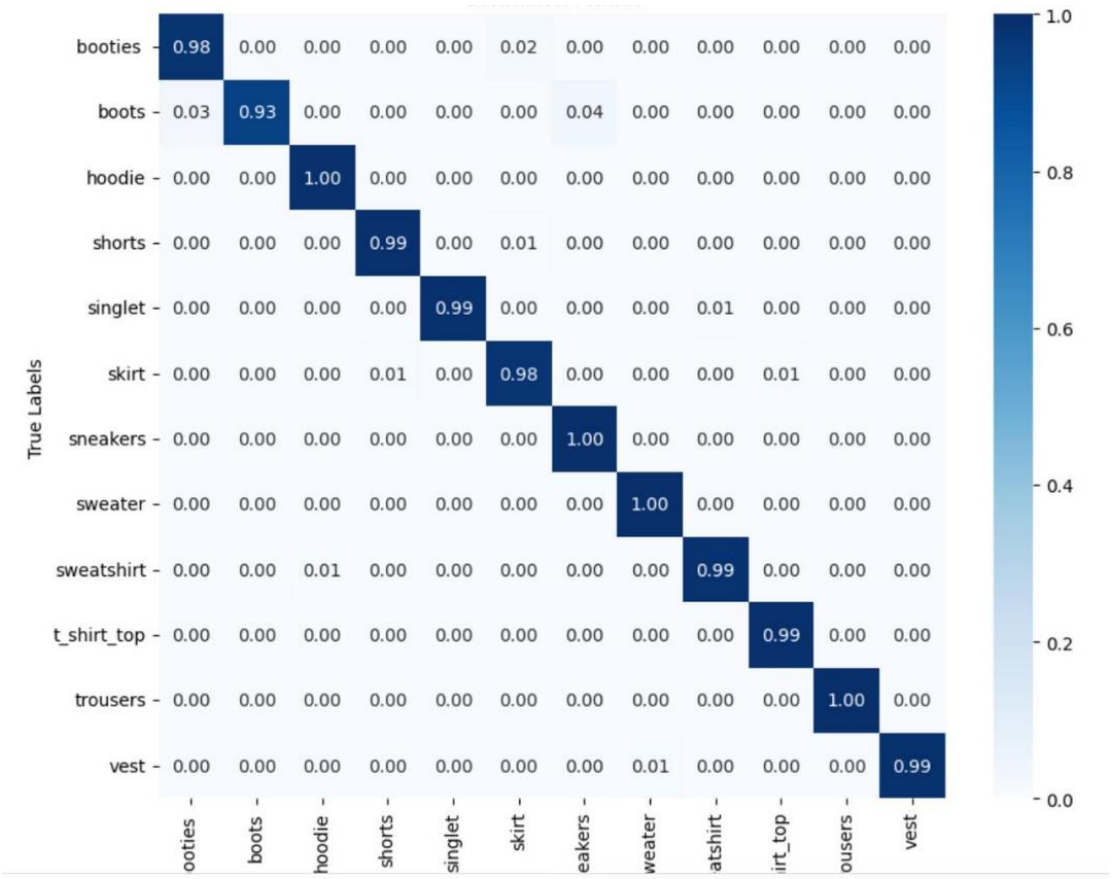

Рисунок 3.31 – Матриця плутанини для архітектури ResNet50V2 у форматі SavedModel

Архітектура MobileNetV2 (рис. 3.32) показала відмінні результати, досягнувши точності класифікації в межах 97%–100% для більшості категорій, таких як жилети, кросівки, шорти та інші. Однак найбільші помилки були зафіксовані при класифікації черевиків, де точність склала 90%. Аналіз матриці плутанини показує, що черевики були помилково класифіковані як спідниці та кросівки.

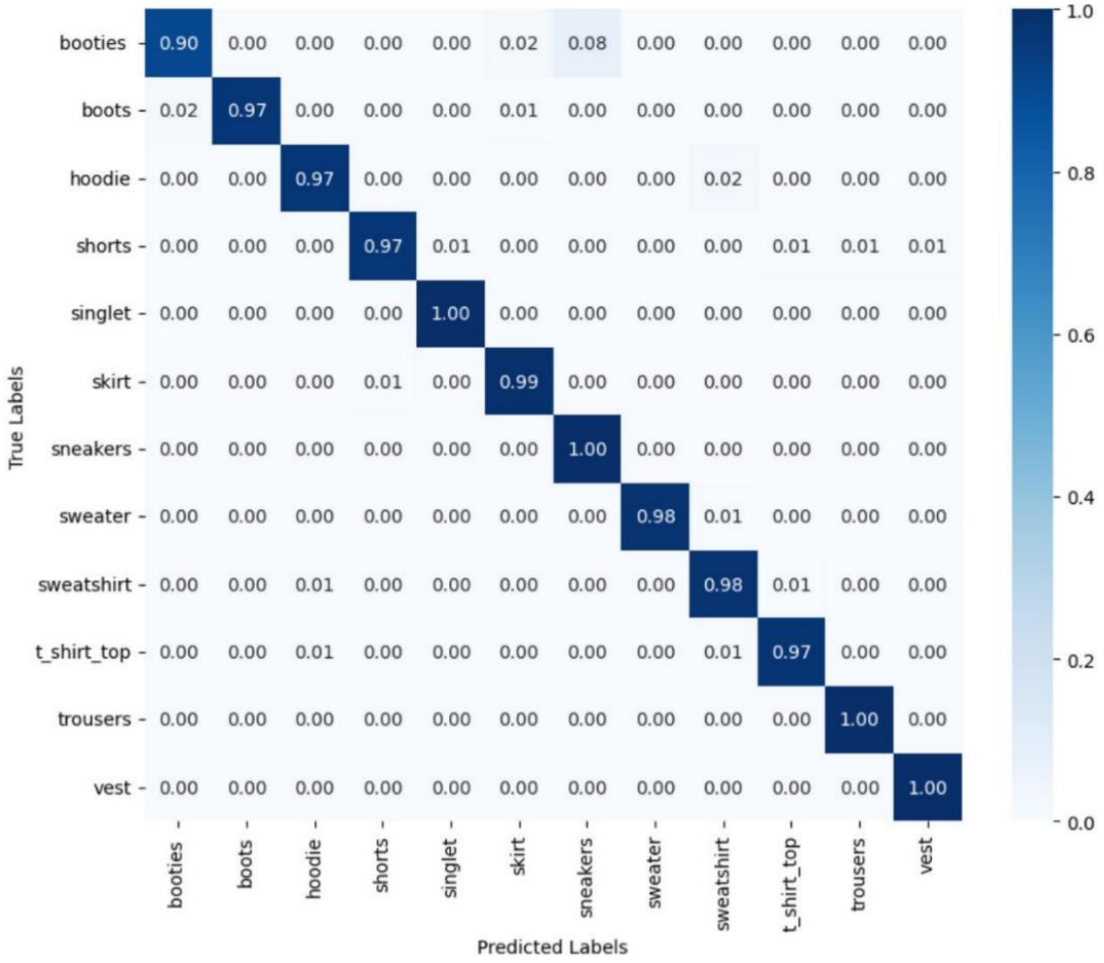

Рисунок 3.32 – Матриця плутанини для архітектури MobileNetV2 у форматі SavedModel

Четвертий етап включає крос-платформне тестування моделей у трьох форматах на прикладі архітектури MobileNetV2. У форматі SavedModel архітектура MobileNetV2 (рис. 3.33) продемонструвала високу точність класифікації, підтверджуючи ефективність обраної архітектури для завдань класифікації одягу. Жилет, худі та кросівки були класифіковані з максимальною точністю 100%, що свідчить про високу впевненість моделі у своїх прогнозах для цих класів. Інші категорії також показали високі результати: спідниця – 98,93%, футболка – з точністю 99,91%, а черевики – з точністю 99,93%. Отримані результати у форматі SavedModel підтверджують здатність моделі ефективно класифікувати різноманітні категорії одягу, демонструючи стабільність та надійність.

Рисунок 3.33 – Тестування архітектури MobileNetV2 у форматі SavedModel

Після конвертації моделі у формат TFLite її продуктивність залишилася стабільною, тобто точність не змінилася (рис. 3.34). Це підтверджує, що модель здатна зберігати свою ефективність навіть після оптимізації для мобільних платформ. Конвертація у формат TFLite забезпечує суттєве скорочення розміру моделі та підвищення швидкості інференсу, зберігаючи при цьому високий рівень точності. Це робить модель придатною для реального використання в умовах обмежених ресурсів, таких як смартфони або вбудовані системи. Зокрема, жилет, худі та кросівки були знову класифіковані з точністю 100%. Інші категорії також продемонстрували такі ж показники: спідниця – 98,93%, футболка – 99,91%, черевики – 99,93%.

Після квантування моделі у формат Quantized TFLite спостерігається невелике зниження точності (рис. 3.35), але MobileNetV2 продовжує демонструвати високі показники. Архітектура MobileNetV2 показала, що спідниця була класифікована з точністю 60,16%, а жилет, худі, кросівки та черевики – 99,61%. Незважаючи на втрату точності для одного з класів, модель зберегла стабільність прогнозів для решти категорій, що підтверджує

доцільність використання квантування для створення легких моделей, орієнтованих на мобільні пристрої або вбудовані системи.

Рисунок 3.34 – Тестування архітектури MobileNetV2 у форматі TFLite

Рисунок 3.35 – Тестування архітектури MobileNetV2 у форматі Quantized TFLite

Загалом, цей етап тестування показав, що моделі є надійними та придатними для застосування на мобільних платформах навіть після конвертації та оптимізації моделей.

Тестування на мобільному застосунку (рис. 3.36) дозволяє перевірити, як модель взаємодіє з інтерфейсом користувача, наскільки швидко вона проводить обробку зображень, а також визначити її точність у різних умовах освітлення, ракурсах та типах зображень.

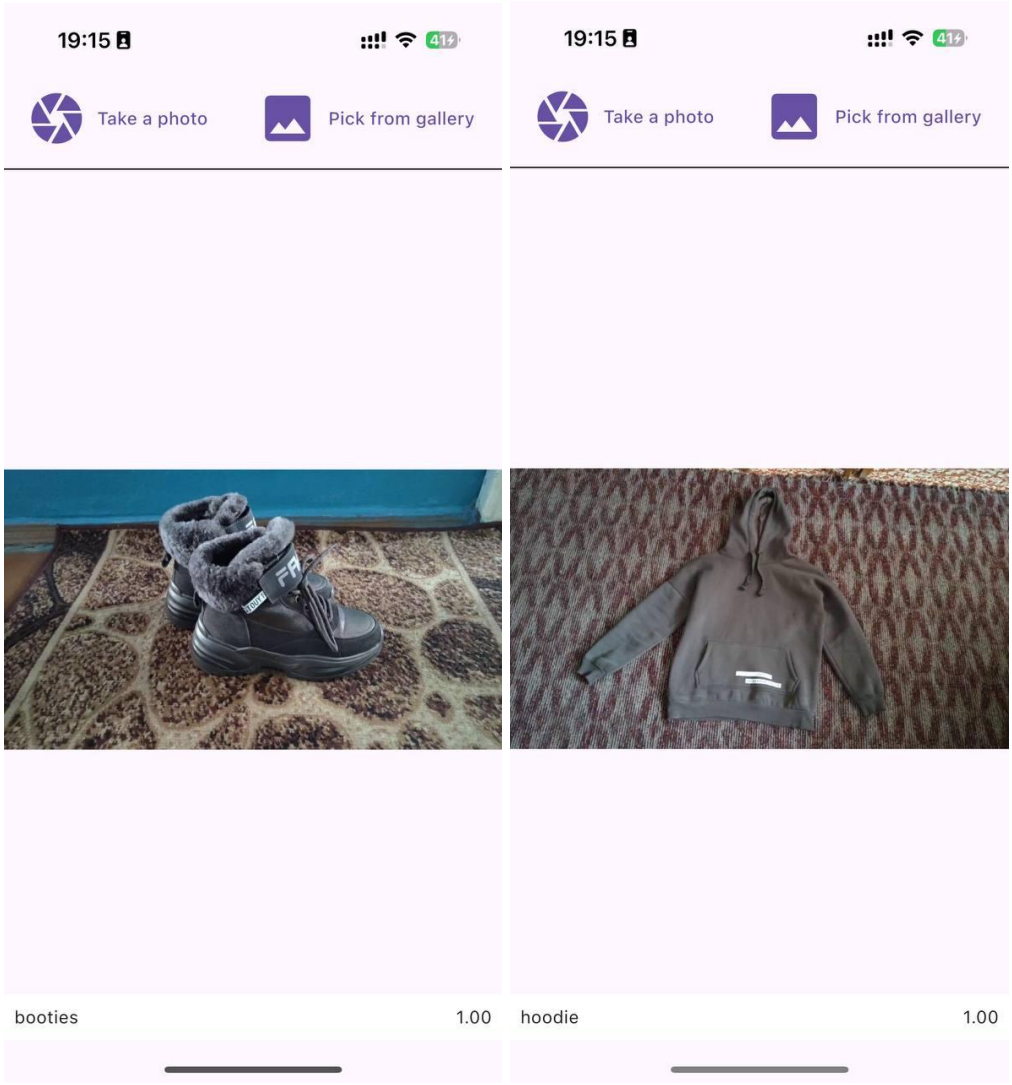

Рисунок 3.36 – Результати роботи моделі EfficientNetB0 в мобільному застосунку

На рисунку 3.36 продемонстровано роботу моделі EfficientNetB0, яка успішно класифікує зображення предметів одягу, таких як черевики та худі, із

впевненістю 100%. Це свідчить про високу точність моделі, адаптованої до реального середовища. Такий підхід до тестування дозволяє виявити можливі слабкі місця або нюанси моделі, які можуть потребувати оптимізації або покращення перед її розгортанням у виробниче середовище.

Після всіх етапів комп'ютерного моделювання можна здійснити загальне порівняння п'яти архітектур нейронних мереж. Всі результати наведені в таблиці 3.1.

Таблиця 3.1 – Оцінка моделей за метриками точності та якості класифікації

Параметри	EfficientNetB0	NASNetMobile	DenseNet121	ResNet50V2	MobileNetV2
1	2	3	4	5	6
Accuracy у форматі SavedModel	0,990	0,967	0,986	0,990	0,980
Precision у форматі SavedModel	0,990	0,967	0,986	0,990	0,980
F1-Score у форматі SavedModel	0,99	0,973	0,986	0,99	0,98
Recall у форматі SavedModel	0,99	0,967	0,986	0,99	0,98
ROC-AUC у форматі SavedModel	0,999	0,999	0,999	0,999	0,999
Log Loss у форматі SavedModel	0,034	0,104	0,06	0,06	0,054
Accuracy у форматі TFLite	0,99	0,967	0,986	0,99	0,98
Precision у форматі TFLite	0,99	0,965	0,988	0,99	0,98
F1-Score у форматі TFLite	0,99	0,973	0,986	0,99	0,98
Recall у форматі TFLite	0,99	0,967	0,986	0,99	0,98
ROC-AUC у форматі TFLite	0,999	0,999	0,999	0,999	0,999
Log Loss у форматі TFLite	0,034	1,677	1,643	1,646	1,647
Accuracy у форматі Quantized TFLite	0,921	0,189	0,269	0,971	0,96
Precision у форматі Quantized TFLite	0,926	0,252	0,367	0,972	0,96

Продовження таблиці 3.1

1	2	3	4	5	6
F1-Score у форматі Quantized TFLite	0,921	0,134	0,227	0,971	0,96
Recall у форматі Quantized TFLite	0,921	0,189	0,269	0,971	0,96
ROC-AUC у форматі Quantized TFLite	0,985	0,598	0,689	0,998	0,998
Log Loss у форматі Quantized TFLite	1,704	2,412	2,339	1,656	1,67

Виходячи з результатів таблиці 3.1, можна зробити висновки, що архітектури ResNet50V2 та EfficientNetB0 є найефективнішими архітектурами для використання у різних форматах, включаючи SavedModel, TFLite та Quantized TFLite. Вони демонструють найвищі показники accuracy, precision, recall, F1-score, ROC-AUC та стабільність, навіть при квантованні, що робить їх оптимальними для застосунків на мобільних пристроях або в умовах обмежених ресурсів.

Архітектура MobileNetV2 демонструє збалансовані результати, особливо у форматі TFLite та Quantized TFLite, залишаючись продуктивною та стабільною. Це робить її ефективною для мобільних та ресурсозалежних задач, хоч вона і поступається ResNet50V2 та EfficientNetB0 за деякими метриками.

Архітектури NASNetMobile та DenseNet121 виявилися менш ефективними у квантованому форматі, але добре працюють у SavedModel та TFLite. Їх можна рекомендувати для застосувань із більшими ресурсами, де квантовані обмеження не є критичними.

Отже, архітектури ResNet50V2 та EfficientNetB0 є пріоритетними виборами для задач з обмеженими ресурсами, тоді як MobileNetV2 забезпечує компроміс між продуктивністю та компактністю. NASNetMobile та DenseNet121 підходять для більш ресурсозалежних середовищ, де важлива висока точність у неквантованих форматах.

ВИСНОВКИ

У даній кваліфікаційній роботі було досліджено та розроблено метод класифікації зображень одягу з використанням сучасних архітектур згорткових нейронних мереж. Основна мета дослідження полягала в аналізі продуктивності та оптимізації моделей для класифікації одягу на основі власного розробленого датасету. Крім того, був розроблений мобільний застосунок для класифікації одягу, на якому також проводився аналіз використання обраних архітектур, що дозволило оцінити їх ефективність у реальних умовах роботи на мобільному пристрої.

Методологія дослідження включала налаштування архітектур згорткових нейронних мереж, таких як MobileNetV2, EfficientNetB0, DenseNet121, NASNetMobile та ResNet50V2. У рамках дослідження застосовувалися техніки оптимізації, включаючи регуляризацію, налаштування гіперпараметрів, аугментацію даних для покращення загальної продуктивності моделей. Моделі оцінювалися за допомогою матриці плутанини та метрик точності, таких як accuracy, precision, recall, F1-score, ROC-AUC та Log Loss. Додатково проводилася візуалізація результатів для детального аналізу роботи моделей.

Дослідження показало, що для задач класифікації зображень одягу на мобільних пристроях з обмеженими ресурсами найефективнішими є архітектури ResNet50V2 та EfficientNetB0, які демонструють високу точність і стабільність навіть після квантування. Архітектура MobileNetV2 є збалансованим вибором для мобільних застосунків, забезпечуючи хорошу продуктивність і компактність, хоч і поступаючись у деяких метриках більш потужним моделям. NASNetMobile та DenseNet121 доцільно використовувати на пристроях з більшими ресурсами у форматах SavedModel та TFLite, де висока точність важливіша за вимоги до оптимізації ресурсів.

Дане дослідження має практичну значущість для дослідників та розробників мобільних застосунків, що вирішують задачі класифікації зображень. Розроблений мобільний застосунок для класифікації одягу дозволяє на практиці використовувати обрані архітектури нейронних мереж та оцінювати їхню ефективність. Також був створений власний датасет зображень одягу, який можуть використовувати інші дослідники та розробники для своїх проєктів, що сприятиме розвитку застосунків у сфері класифікації та автоматичного підбору одягу.

Наукова новизна даного дослідження полягає в розробці методу класифікації зображень одягу, що використовує сучасні архітектури нейронних мереж, які спеціально адаптовані для роботи на мобільних пристроях. Під час виконання роботи був створений унікальний датасет із зображеннями одягу, який може бути корисним для інших дослідників та розробників. Також був розроблений мобільний застосунок, який надав можливість перевірити моделі безпосередньо на мобільному пристрої.

Подальші напрями дослідження передбачають розширення датасету додатковими категоріями одягу, що дозволить моделям краще адаптуватися до ширшого спектра зображень та дослідження можливостей використання інших архітектур нейронних мереж для підвищення точності класифікації.

Результати дослідження апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [50].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

2. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

3. Daradkeh, Y.I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L.A., and Ahmad, N. (2021) Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, *IEEE Access*, 9, pp. 13417-13428.

4. Zhang, X., Wang, Y., Liang, D., & Zhang, Y. (2020). A review of clothing recognition methods using deep learning and its application. *Multimedia Tools and Applications*, *79*(19-20), 13929-13947.

5. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

6. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

7. Зінченко, О. В., Звенігородський, О. С., & Кисіль, Т. М. (2022). Згорткові нейронні мережі для вирішення задач комп'ютерного зору. *Телекомунікаційні та інформаційні технології*, (2), 4-12.

8. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

9. Deb, P., Kar, N., Hassan, K. L., & Biswas, B. (2024). Advanced copy-move forgery detection: utilizing AKAZE in conjunction with SIFT algorithm for image forensics. *Microsystem Technologies*, 1-9.

10. Bouchene, M. M. (2024). Bayesian optimization of histogram of oriented gradients (HOG) parameters for facial recognition. *The Journal of Supercomputing*, 1-32.

11. Peer, M., Kleber, F., & Sablatnig, R. (2024, August). SAGHOG: Self-supervised Autoencoder for Generating HOG Features for Writer Retrieval. In *International Conference on Document Analysis and Recognition* (pp. 121-138). Cham: Springer Nature Switzerland.

12. Колосніченко, М. В., & Процик, К. Л. (2011). Мода і одяг. Основи проектування та виробництва одягу.

13. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, *Telecommunications and Radio Engineering*, 79(10), pp. 855-863.

14. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua,* 73(3), pp. 6069-6084.

15. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, vol. 11, pp. 126938-126949.

16. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

17. Cychnerski, J., Brzeski, A., Boguszewski, A., Marmolowski, M., & Trojanowicz, M. (2017, September). Clothes detection and classification using convolutional neural networks. In *2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA)* (pp. 1-8). IEEE.

18. TSURU, T., SUGAHARA, M., & NISHIMURA, H. (2019). A classification method for silhouettes of various clothes. In *International Symposium on Affective Science and Engineering ISASE2019* (pp. 1-4). Japan Society of Kansei Engineering.

19. Tan, Z., Hu, Y., Luo, D., Hu, M., & Liu, K. (2020). The clothing image classification algorithm based on the improved Xception model. *International Journal of Computational Science and Engineering*, *23*(3), 214-223.

20. Liang, X., Lin, L., Yang, W., Luo, P., Huang, J., & Yan, S. (2016). Clothes co-parsing via joint image segmentation and labeling with application to clothing retrieval. *IEEE Transactions on Multimedia*, *18*(6), 1175-1186.

21. Shin, S. Y., Jo, G., & Wang, G. (2023). A novel method for fashion clothing image classification based on deep learning. *Journal of Information and Communication Technology*, *22*(1), 127-148.

22. Babuc, D., & Fortiş, A. E. (2024, April). Fine-Tuned CNN for Clothing Image Classification on Mobile Edge Computing. In *International Conference on Advanced Information Networking and Applications* (pp. 65-75). Cham: Springer Nature Switzerland.

23. Jeong, H. S., Lee, S. Y., & Lee, C. K. (2024). Deep learning-based clothing attribute classification using fashion image data. *Smart Media Journal*, *13*(4), 57-64.

24. Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, *35*(5), 1285-1298.

25. Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS journal of photogrammetry and remote sensing*, *173*, 24-49.

26. Chua, L. O., & Roska, T. (1993). The CNN paradigm. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, *40*(3), 147-156.

27. Тітов, С. В., Тітова, О. В., & Чорна, О. С. (2023). Метод знаходження апроксимацій приблизних множин з використанням систем числення. *Системи обробки інформації*, *2*(173), 58–62.

28. Dablain, D., Jacobson, K. N., Bellinger, C., Roberts, M., & Chawla, N. V. (2024). Understanding CNN fragility when learning with imbalanced data. *Machine Learning*, *113*(7), 4785-4810.

29. Akter, A., Nosheen, N., Ahmed, S., Hossain, M., Yousuf, M. A., Almoyad, M. A. A., ... & Moni, M. A. (2024). Robust clinical applicable CNN and U-Net based algorithm for MRI classification and segmentation for brain tumor. *Expert Systems with Applications*, *238*, 122347.

30. Varone, G., Boulila, W., Driss, M., Kumari, S., Khan, M. K., Gadekallu, T. R., & Hussain, A. (2024). Finger pinching and imagination classification: A fusion of CNN architectures for IoMT-enabled BCI applications. *Information Fusion*, *101*, 102006.

31. Zhang, F., Yin, J., Wu, N., Hu, X., Sun, S., & Wang, Y. (2024). A dual-path model merging CNN and RNN with attention mechanism for crop classification. *European Journal of Agronomy*, *159*, 127273.

32. Singh, M. K. (2024). A text independent speaker identification system using ANN, RNN, and CNN classification technique. *Multimedia Tools and Applications*, *83*(16), 48105-48117.

33. Тітов, С. В., Тітова, О. В., & Чорна, О. С. (2022). Опис нескоротних наборів ознак в приблизних множинах з використанням систем числення. *Збірник наукових праць Харківського національного університету Повітряних Сил*, *1*(71), 106–110.

34. Laurer, M., Van Atteveldt, W., Casas, A., & Welbers, K. (2024). Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and BERT-NLI. *Political Analysis*, *32*(1), 84-100.

35. Chen, H., Tao, R., Zhang, H., Wang, Y., Li, X., Ye, W., ... & Savvides, M. (2024). Conv-adapter: Exploring parameter efficient transfer learning for

convnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1551-1561).

36. Ahn, K., Zhang, Z., Kook, Y., & Dai, Y. (2024). Understanding Adam optimizer via online learning of updates: Adam is FTRL in disguise. *arXiv preprint arXiv:2402.01567*.

37. Kabiri, H., Ghanou, Y., Khalifi, H., & Casalino, G. (2024). AMAdam: adaptive modifier of Adam method. *Knowledge and Information Systems*, 1-32.

38. Gurin, D., Yevsieiev, V., Maksymova, S., & Abu-Jassar, A. (2024). Effect of Frame Processing Frequency on Object Identification Using MobileNetV2 Neural Network for a Mobile Robot. *Multidisciplinary Journal of Science and Technology*, *4*(8), 36-44.

39. Ashtagi, R., Kharat, P. V., Sarmalkar, V., Hosmani, S., Patil, A. R., Akkalkot, A. I., & Padthe, A. (2024). Enhancing melanoma skin cancer diagnosis through transfer learning: An EfficientNetb0 approach. *Acadlore Trans. Mach. Learn*, *3*(1), 57-69.

40. Potsangbam, J., & Devi, S. S. (2024). Classification of Breast Cancer Histopathological Images Using Transfer Learning with DenseNet121. *Procedia Computer Science*, *235*, 1990-1997.

41. Riyadi, S., Abidin, F. A., & Audita, N. (2024, May). Comparison of ResNet50V2 and MobileNetV2 Models in Building Architectural Style Classification. In *2024 International Conference on Intelligent Systems and Computer Vision (ISCV)* (pp. 1-8). IEEE.

42. Aksoy, İ., & Adem, K. (2024). Optimizing hyperparameters for enhanced performance in convolutional neural networks: A study using NASNetMobile and DenseNet201 Models. *Mühendislik Bilimleri ve Araştırmaları Dergisi*, *6*(1), 42-52.

43. Vanacore, A., Pellegrino, M. S., & Ciardiello, A. (2024). Fair evaluation of classifier predictive performance based on binary confusion matrix. *Computational Statistics*, *39*(1), 363-383.

44. Karunarathna, I., Gunasena, P., Hapuarachchi, T., & Gunathilake, S. (2024). Comprehensive data collection: Methods, challenges, and the importance of accuracy.

45. Liu, Y., Li, Y., & Xie, D. (2024). Implications of imbalanced datasets for empirical ROC-AUC estimation in binary classification tasks. *Journal of Statistical Computation and Simulation*, *94*(1), 183-203.

46. Zhang, Z., Shi, L., & Zhou, D. X. (2024). Classification with deep neural networks and logistic loss. *Journal of Machine Learning Research*, *25*(125), 1-117.

47. Porter, L., & Zingaro, D. (2024). *Learn AI-assisted Python Programming*. Simon and Schuster.

48. De Santana, T. L., Neto, P. A. D. M. S., De Almeida, E. S., & Ahmed, I. (2024). Bug Analysis in Jupyter Notebook Projects: An Empirical Study. *ACM Transactions on Software Engineering and Methodology*, *33*(4), 1-34.

49. Kuroki, M. (2023). Integrating data science into an econometrics course with a Kaggle competition. *The Journal of Economic Education*, *54*(4), 364-378.

50. Шкарупа, А. О. (2024). Аналіз сучасних тенденцій у сфері технологій класифікації зображень.