МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ШТИНДА Віктор Володимирович

Модель розпізнавання їжі на основі розпізнавання образів / Model for food recognition based on image recognition

спеціальність: 122 - Комп'ютерні науки

освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21

В.В. Штинда

_____

Науковий керівник:

к.т.н., доцент

Х.В. Ліп'яніна-Гончаренко

_____

Кваліфікаційну роботу

допущено до захисту:

«___» _____ 20___ р.

В.о. завідувача кафедри

_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ - 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Штинда Віктор Володимирович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Модель розпізнавання їжі на основі розпізнавання образів / Model for food recognition based on image recognition

керівник роботи к.т.н., доцент Х.В. Ліп'яніна- Гончаренко
затверджені наказом по університету від 20 грудня 2024 року № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити

- огляд предметної області та аналіз задачі автоматичного розпізнавання їжі у системах комп'ютерного зору;
- аналіз класичних методів і підходів до класифікації зображень;
- огляд сучасних згорткових нейронних мереж і глибинних моделей для класифікації зображень їжі;
- постановка задачі двокласової класифікації зображень їжі;
- розробка базової згорткової нейронної мережі типу Tiny-VGG;
- дослідження та застосування підходів трансферного навчання для покращення якості розпізнавання;
- реалізація моделі, моделювання процесу навчання та проведення експериментальних досліджень;
- аналіз і порівняння отриманих результатів за основними метриками якості класифікації.

5. Перелік графічного матеріалу у роботі

- схема архітектури згорткової нейронної мережі типу Tiny-VGG для задачі розпізнавання зображень їжі;
- графіки порівняльного аналізу ефективності базової та покращених

моделей розпізнавання зображень їжі за основними метриками якості.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 1.12.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2025 р.	

Студент _____ В.В. Штинда
підпис

Керівник роботи _____ к.т.н., доцент Х.В. Ліп’яніна- Гончаренко

РЕЗЮМЕ

Кваліфікаційна робота на тему «Модель розпізнавання їжі на основі розпізнавання образів» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп’ютерні науки» освітньо-професійної програми «Комп’ютерні науки» виконана на 64 сторінках і містить 22 ілюстрації, 1 додаток та 42 використані джерела.

Мета роботи — розроблення, реалізація та експериментальне обґрунтування моделі автоматичного розпізнавання зображень їжі на основі згорткових нейронних мереж із використанням компактної архітектури Tiny-VGG та підходів трансферного навчання для забезпечення стабільної якості класифікації за обмежених обчислювальних ресурсів.

Результати дослідження: проєктовано та реалізовано згорткову нейронну мережу типу Tiny-VGG для двокласової класифікації зображень їжі (pizza/steak), що поєднує стандартизовану попередню обробку, керовану аугментацію даних і ефективний протокол навчання. Проведено порівняльний аналіз конфігурацій із різними швидкостями навчання, експериментальне моделювання та оцінювання якості за метриками accuracy, precision, recall, F1 та ROC-AUC. Отримані результати підтверджують доцільність використання компактної CNN-архітектури для задач розпізнавання їжі та демонструють високу узагальнювальну здатність моделі.

Практичне значення роботи полягає у можливості застосування розробленої моделі в мобільних застосунках харчових щоденників, системах рекомендацій у закладах харчування, сервісах контролю якості та інших інформаційних системах комп’ютерного зору, де важливими є швидкість інференсу, відтворюваність та помірні вимоги до апаратних ресурсів.

Ключові слова: КОМП’ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ЇЖІ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, TINY-VGG, ГЛИБИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, ТРАНСФЕРНЕ НАВЧАННЯ, АУГМЕНТАЦІЯ ДАНИХ, МЕТРИКИ ЯКОСТІ.

ABSTRACT

The qualification thesis entitled “Model for Food Recognition Based on Image Recognition” for obtaining the Master’s degree in specialty 122 Computer Science of the educational and professional program Computer Science is completed on 64 pages and contains 22 figures, 1 appendix, and 42 references.

The aim of the thesis is the development, implementation, and experimental validation of an automatic food image recognition model based on convolutional neural networks using a compact Tiny-VGG architecture and transfer learning approaches to ensure stable classification performance under limited computational resources.

Research results: a Tiny-VGG–based convolutional neural network for binary food image classification (pizza/steak) has been designed and implemented, combining standardized preprocessing, controlled data augmentation, and an efficient training protocol. A comparative analysis of configurations with different learning rates was conducted, along with experimental modeling and performance evaluation using accuracy, precision, recall, F1-score, and ROC-AUC metrics. The obtained results confirm the effectiveness of a compact CNN architecture for food recognition tasks and demonstrate high generalization capability of the proposed model.

The practical significance of the thesis lies in the possibility of applying the developed model in mobile food diary applications, recommendation systems in food services, quality control services, and other computer vision–based information systems, where inference speed, reproducibility, and moderate hardware requirements are critical.

Keywords: COMPUTER VISION, FOOD IMAGE RECOGNITION, CONVOLUTIONAL NEURAL NETWORK, TINY-VGG, DEEP LEARNING, IMAGE CLASSIFICATION, TRANSFER LEARNING, DATA AUGMENTATION, QUALITY METRICS.

ЗМІСТ

ВСТУП

Актуальність теми. Автоматичне розпізнавання їжі на зображеннях є затребуваною задачею комп’ютерного зору з прямими застосуваннями у мобільних системах харчового щоденника, клінічному моніторингу дієти, сервісах рекомендацій у закладах харчування, роботизованому сортуванні та контролі якості. Прорив глибинного навчання, зокрема згорткових нейромереж (CNN) та підходів трансферного навчання, забезпечив суттєве зростання точності класифікації у відкритих доменах, однак у вузьких доменах на кшталт «food» все ще існують виклики: невеликі навчальні вибірки, зміна умов зйомки й композиції, сильний фон і доменний зсув відносно ImageNet.

Сучасні підходи до класифікації зображень їжі базуються на CNN-архітектурах типу VGG, ResNet, Inception, MobileNet, EfficientNet, а також на практиках transfer learning із частковим донавчанням верхніх шарів. Для підвищення узагальнюваності застосовують керовані аугментації (геометричні, фотометричні, міксувальні), регуляризацію (Dropout, L2, label smoothing), стратифіковане розбиття даних і оцінювання за accuracy, F1, ROC-AUC. У роботі враховано ці напрацювання та адаптовано їх до двокласової постановки («pizza»/«steak») із фокусом на прозорій, відтворюваній архітектурі Tiny-VGG і порівнянні з варіантом transfer learning.

Мета дослідження — розробити, реалізувати та експериментально обґрунтувати модель розпізнавання їжі на основі розпізнавання образів з використанням компактної CNN-архітектури (Tiny-VGG) і/або трансферного навчання, забезпечивши відтворюваність протоколу та стабільну якість на відкладених даних.

Для досягнення мети передбачено розв’язання таких завдань:

1. виконати аналітичний огляд методів і інструментів класифікації food-зображень;
2. сформулювати постановку задачі, вимоги до даних і метрик;
3. спроєктувати концептуальну схему обробки даних та архітектури моделі;

4. реалізувати базову CNN (Tiny-VGG) і варіант трансферного навчання;
5. налаштувати протокол навчання (оптимізатор, графік швидкості навчання, рання зупинка, класові ваги);
6. провести моделювання, побудувати криві навчання, матриці помилок, ROC-криві;
7. виконати тестування на індивідуальних зразках і батч-прикладах;
8. інтерпретувати результати та сформулювати рекомендації щодо подальшого вдосконалення.

Об’єкт дослідження — процес автоматичної класифікації зображень їжі у системах комп’ютерного зору.

Предмет дослідження — моделі та алгоритми розпізнавання образів (CNN, transfer learning), а також протоколи підготовки даних і оцінювання якості, що визначають узагальнюваність у двокласовій постановці.

Методи дослідження. Теоретичною основою є методи глибинного навчання (згорткові мережі, функції втрат бінарної класифікації, регуляризація), статистична перевірка якості (accuracy, precision/recall/F1, ROC-AUC), крос-валідація та бутстреп-оцінки довірчих інтервалів. На етапі інженерії даних застосовано нормалізацію, уніфікацію розміру, керовані аугментації; на етапі оптимізації — алгоритми Adam/SGD із розкладами швидкості навчання та ранньою зупинкою.

Наукова новизна полягає у поєднанні легкої, інтерпретованої архітектури Tiny-VGG із продуманим мінімалістичним протоколом аугментацій і регуляризації для вузького food-домена, а також у методичному зіставленні «навчання з нуля» та transfer learning за однакових умов підготовки даних.

Практичне значення результатів. Реалізована модель може бути інтегрована до мобільних застосунків оцінювання калорійності, інформаційних кіосків у закладах харчування, модулів контролю асортименту/якості в рітейлі. Публікація скриптів інференсу та збережених ваг забезпечує швидке розгортання у середовищах із обмеженими ресурсами, а прозора архітектура спрощує подальше донавчання під локальні меню.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися на студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025), яка відбулася в місті Тернополі 27–29 травня 2025 року та IX Міжнародної студентської наукової конференції «Модернізація та сучасні українські і світові наукові дослідження» 14 листопада 2025 в місті Житомир, Україна.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ, АЛГОРИТМІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ЇЖІ

1.1 Базові архітектури типу VGG/Tiny-VGG і підходи до трансферного навчання

Архітектури сімейства VGG належать до класичних згорткових мереж, у яких глибинність досягається послідовним накопиченням однакових за формою блоків із малими ядрами 3×3і періодичним субсемплінгом за допомогою max-pooling [1]. Така конструкція історично продемонструвала сильну здатність до виділення локальних візуальних патернів на великих корпусах даних, зокрема ImageNet [2], і стала «робочою конячкою» для численних задач класифікації зображень. Модифікації на кшталт Tiny-VGG зменшують число каналів і пов’язану з цим кількість параметрів, що дозволяє навчати моделі на невеликих наборах даних і на обмежених апаратних ресурсах без критичної втрати якості [1, 3].

Ключовою ідеєю VGG є заміна великих згорток на послідовності менших: дві послідовні 3×3згортки мають ефективне рецептивне поле 5×5, три — 7×7, але за меншої кількості параметрів і з більшою нелінійністю між ними завдяки ReLU [1]. Це сприяє кращій апроксимаційній здатності та стабільнішій оптимізації в порівнянні з одноетапними великими ядрами [4]. Batch Normalization і Dropout, хоч і не були базовою частиною початкової VGG, широко застосовуються в похідних реалізаціях для поліпшення стабільності градієнтів та узагальнюваності [5, 6].

У контексті невеликих доменно-специфічних задач (як-от класифікація страв за зображеннями) архітектури типу Tiny-VGG вигідні завдяки компромісу між «виразною силою» і контролем перенавчання. Обмежена кількість фільтрів у шарах знижує варіативність моделі, що при малих навчальних множинах часто трансформується у кращу валідовану точність [3, 7]. Разом із цим доцільно застосовувати ранню зупинку та регуляризацію ваг (L2/weight decay) [6], [8].

Трансферне навчання стало де-факто стандартом, коли цільовий датасет недостатньо великий для тренування «з нуля». Переднавчені на ImageNet екстрактори ознак (VGG-16/19, ResNet, Inception, EfficientNet тощо) формують репрезентації, які добре переносяться на різні візуальні домени [2], [9]–[11]. Ранньокаскадні шари фіксують універсальні низькорівневі патерни (градієнти, текстури), тоді як пізні — більш доменно-залежні [12]. Тому типовою є стратегія «заморожування» ранніх шарів і тонкого донавчання (fine-tuning) верхніх [12, 13].

У трансферному навчанні розрізняють два базові режими: (а) feature extraction — коли всі згорткові шари фіксуються, а навчається лише новий класифікатор над ними; (б) fine-tuning — коли частина верхніх згорток розморожується й донавчається разом з класифікатором [9, 12]. Вибір режиму залежить від обсягу та «схожості» даних: за відсутності великої кількості маркованих зразків або суттєвій відмінності домену безпечніше починати з feature extraction, поступово переходячи до fine-tuning невеликої кількості верхніх шарів [9, 11, 14].

Емпіричні дослідження демонструють, що якісніші моделі на ImageNet не завжди однаково добре переносяться на всі домени; однак у середньому кореляція між ImageNet-якістю і переносимістю позитивна [11]. Додаткові фактори — нормалізація входу, вибір scheduler’а швидкості навчання, баланс класів, — часто визначають різницю між «хорошим» і «відмінним» переносом [10, 15, 16].

Практика discriminative learning rates (різні ηдля різних шарів) дає виграш при fine-tuning: глибокі «заморожені» шари оновлюються дуже обережно, тоді як нові верхні — агресивніше [14, 17]. Сучасні графіки η(cosine annealing з рестартами, циклічні LR) сприяють кращому виходу з плато та зниженню валідованої помилки [16, 18]. Для стабільності в невеликих задачах зручно комбінувати Adam на ранніх етапах і SGD з momentum у фазі доведення [8, 16, 19].

VGG-подібні сітки, попри поважний вік, зберігають актуальність як «прозора» база для освітніх і прикладних сценаріїв, де важливі інтерпретованість і простота відлагодження. Вони часто використовуються як стартова точка порівняння з більш сучасними архітектурами (ResNet, Inception-v3, MobileNetV2, EfficientNet), які завдяки залишковим з'єднанням чи пошуку архітектур за критеріями ефективності досягають кращого співвідношення «якість/обчислення» [4, 10], [20]–[22].

Для задачі «піца vs стейк» доцільно розглядати дві траєкторії: (1) Tiny-VGG, навчена з нуля з інтенсивною аугментацією та регуляризацією; (2) легка переднавчена модель (наприклад, MobileNetV2/EfficientNet-B0) у режимі feature extraction із тонким донавчанням останніх блоків [20, 21]. Емпірично другий підхід зазвичай дає вищу стабільність валідованої якості на невеликому корпусі [9, 11, 21].

У випадках вираженого доменного зсуву (food-фото часто відрізняються від ImageNet за композицією, кольорами та фактурою) корисними є адаптаційні прийоми: перепідбір нормалізаційних констант, попереднє донавчання на суміжних food-датасетах (Food-101/UEC-Food) або застосування self-supervised/weakly-supervised переднавчання, що зменшує залежність від лейблів [1], [23]–[25].

1.2 Аугментація даних і практики забезпечення узагальнюваності

Аугментація даних — це систематичне застосування стохастичних, label-preserving перетворень, покликаних збільшити різноманітність навчальних прикладів і зменшити розрив між тренувальним і тестовим розподілами [26]. Для food-зображень характерні варіації масштабу, обертання, зсувів, освітлення та фону; тому базовий набір перетворень включає невеликі повороти, кропи/зум, горизонтальні віддзеркалення, зсуви та легкі фотометричні зміни [26, 27].

З математичного погляду аугментацію можна розглядати як стохастичний оператор T, що породжує вибірки $\tilde{x} = T(x)$так, що умовний розподіл $p(y \mid x)$зберігається, а підтримка $p(x)$розширюється в околі реальних спостережень. Це індукує усереднення рішень моделі на орбіті перетворень, що зменшує варіанс оцінки та підвищує робастність до спотворень у тесті [26].

Окрім класичних геометричних і фотометричних трансформацій, сучасні методи пропонують композиційні політики. AutoAugment шукає (засобами підкріплювального навчання) оптимальні комбінації операцій і їх інтенсивностей для заданого датасета [28]; Fast AutoAugment пришвидшує пошук за допомогою займистих евристик [29]; RandAugment відмовляється від пошуку, керуючись лише двома гіперпараметрами (кількість і сила перетворень) [30]. Для невеликих корпусів RandAugment часто дає найкращий компроміс «якість/простота» [30].

Регуляризаційні відсікальні та міксувальні методи — Cutout, Random Erasing, Mixup, CutMix — працюють не як класична симуляція геометрії сцени, а як індукування часткової невизначеності й стимулювання моделі спиратися на глобальні, а не локальні «чит-патерни» [31]–[34]. Теоретично Mixup реалізує випукле змішування прикладів і міток, що еквівалентне лінійній регуляризації задачі, пом’якшуючи межі класифікації [33]. Для кулінарних сцен CutMix/Random Erasing добре маскують «сковороду/тарілку» та інші контексти, зменшуючи спокусу моделі «вчитись на фоні» [32, 34].

Ще один клас — AugMix, який генерує ансамблі слабких перетворень з узгодженням передбачень (consistency loss) і демонструє зростання робастності до корупцій (blur, noise, digital artifacts) без падіння на чистих даних [35]. Для малих датасетів це часто краща альтернатива жорстким фотометричним «атакам», що можуть зруйнувати семантику.

Практики забезпечення узагальнюваності не обмежуються аугментацією. Регуляризація (L2/weight decay), Dropout, Label Smoothing і рання зупинка прямо контролюють складність гіперплощини рішення й чутливість до шуму анотацій

[6, 8, 36, 37]. Для бінарних задач на кшталт «піца/стейк» Label Smoothing ($\varepsilon \in [0.05; 0.1]$) пом'якшує надмірну впевненість і покращує AUC на валідації [37].

Важливу роль відіграє балансування класів: при дисбалансі застосовують класові ваги, стратефікований семплінг, а на етапі оцінювання — макро-F1 і ROC-AUC замість простої accuracy [8, 38]. Для підвищення стабільності оцінки варто використовувати бутстреп-довірчі інтервали або k-fold cross-validation [39].

У процесі навчання доцільні scheduler'и швидкості навчання: cosine annealing (SGDR) і циклічні LR (CLR) довели ефективність як методи, що водночас виконують роль неявної регуляризації, допомагаючи уникати локальних мінімумів [16, 18]. На практиці часто спрацьовує комбінація: фіксована η у фазі «розігріву», далі — cosine/step decay з ReduceLROnPlateau [16, 18, 19].

Test-time augmentation (TTA) — інференс із усередненням передбачень по набору простих перетворень (flips, crops) — забезпечує невеликий, але стабільний виграш без зміни навчальної процедури [26]. Для food-сцен TTA особливо корисний, коли під час зйомки виникають сильні відхилення композиції/масштабу.

1.3 Постановка задачі

Актуальність теми. Автоматичне розпізнавання їжі на зображеннях є затребуваною задачею комп'ютерного зору з прямими застосуваннями у мобільних системах харчового щоденника, клінічному моніторингу дієти, сервісах рекомендацій у закладах харчування, роботизованому сортуванні та контролі якості. Прорив глибинного навчання, зокрема згорткових нейромереж (CNN) та підходів трансферного навчання, забезпечив суттєве зростання точності класифікації у відкритих доменах, однак у вузьких доменах на кшталт

«food» все ще існують виклики: невеликі навчальні вибірки, зміна умов зйомки й композиції, сильний фон і доменний зсув відносно ImageNet.

Сучасні підходи до класифікації зображень їжі базуються на CNN-архітектурах типу VGG, ResNet, Inception, MobileNet, EfficientNet, а також на практиках transfer learning із частковим донавчанням верхніх шарів. Для підвищення узагальнюваності застосовують керовані аугментації (геометричні, фотометричні, міксувальні), регуляризацію (Dropout, L2, label smoothing), стратифіковане розбиття даних і оцінювання за accuracy, F1, ROC-AUC. У роботі враховано ці напрацювання та адаптовано їх до двокласової постановки («pizza»/«steak») із фокусом на прозорій, відтворюваній архітектурі Tiny-VGG і порівнянні з варіантом transfer learning.

Додаткову складність формують тонкі міжкласові відмінності та висока внутрішньокласова варіативність (різні способи подачі страв, наявність соусів, гарнірів, посуду), часткові оклюзії, тіні й різнорідність джерел освітлення. Реальні дані часто містять дисбаланс класів і «шумні» мітки, що погіршує стабільність навчання та веде до переадаптації на вузькі патерни фону. Тому актуальними стають ретельно налаштовані аугментації, контроль складності моделі, механізми ранньої зупинки та інваріантні до фону стратегії попередньої обробки (уніфікація розміру, нормалізація інтенсивностей).

Важливим є питання ефективності: для практичних сценаріїв (мобільні пристрої, системи «на краю» мережі) потрібні компактні моделі з низькою затримкою інференсу та помірними обчислювальними вимогами. Архітектури на кшталт Tiny-VGG забезпечують прийнятний компроміс між якістю й ресурсомісткістю, а transfer learning дає змогу досягати вищої якості за рахунок використання попередньо навчених представлень, коли доступ до великих доменних датасетів обмежений. Додатково, для наукової та прикладної цінності результатів необхідні відтворювані протоколи (фіксація seed-значень, версій бібліотек і апаратних конфігурацій), стандартизовані метрики та прозоре логування кривих навчання.

З огляду на суспільну значущість задачі (здорове харчування, персоналізовані рекомендації, автоматизований контроль якості в рітейлі та громадському харчуванні), дослідження спрямовано на формування методично вивіреної моделі з чіткими інженерними й статистичними гарантіями. Передбачено акцент на інтерпретованості (візуалізація важливих регіонів, наприклад Grad-CAM), етичних аспектах (конфіденційність користувацьких зображень, недискримінаційність алгоритмів) і відкритості результатів (опис пайплайна, параметрів навчання, збережені ваги), що забезпечує можливість подальшого масштабування системи на багатокласові сценарії та доменну адаптацію під локальні умови зйомки.

У підсумку, актуальність зумовлена поєднанням практичної корисності, наукових викликів (тонкі відмінності класів, доменний зсув, дефіцит даних) і потреби у відтворюваних рішеннях, придатних до розгортання в ресурсно обмежених середовищах. Запропонована постановка — двокласова класифікація «pizza/steak» на базі компактної CNN з керованими аугментаціями та можливістю трансферного навчання — слугує репрезентативним полігоном для відпрацювання методів, що в подальшому можуть бути узагальнені на ширший спектр страв і виробничих завдань контролю якості.

Мета дослідження — розробити, реалізувати та експериментально обґрунтувати модель розпізнавання їжі на основі розпізнавання образів з використанням компактної CNN-архітектури (Tiny-VGG) і/або трансферного навчання, забезпечивши відтворюваність протоколу та стабільну якість на відкладених даних.

Для досягнення мети передбачено розв’язання таких завдань:

1. виконати аналітичний огляд методів і інструментів класифікації food-зображень;
2. сформулювати постановку задачі, вимоги до даних і метрик;
3. спроєктувати концептуальну схему обробки даних та архітектури моделі;
4. реалізувати базову CNN (Tiny-VGG) і варіант трансферного навчання;

5. налаштувати протокол навчання (оптимізатор, графік швидкості навчання, рання зупинка, класові ваги);
6. провести моделювання, побудувати криві навчання, матриці помилок, ROC-криві;
7. виконати тестування на індивідуальних зразках і батч-прикладах;
8. інтерпретувати результати та сформулювати рекомендації щодо подальшого вдосконалення.

Висновки до розділу 1

1. У першому розділі здійснено систематизацію підходів до класифікації зображень їжі та обґрунтовано вибір компактної VGG-подібної архітектури (Tiny-VGG) як прозорої та відтворюваної бази для дослідження. Теоретичний аналіз показав, що поєднання малих згорток 3×3, помірної кількості фільтрів і регулярної субдискретизації MaxPool забезпечує достатню виразну силу для двокласових задач із відносно невеликими вибірками, мінімізуючи ризик перенавчання та обчислювальні витрати. Узагальнено переваги та обмеження transfer learning; визначено доцільні режими використання переднавчених екстракторів (feature extraction, fine-tuning) залежно від обсягу даних і близькості домену.

2. Окрема увага приділена аугментації як ключовому механізму підвищення узагальнюваності: наведено доцільні геометричні та фотометричні перетворення для food-сцен, а також сучасні міксувальні/відсікальні практики (mixup/cutout/cutmix) у ролі неявної регуляризації. Визначено роль ранньої зупинки, L2-штрафу, label smoothing, класових ваг і стратифікованих протоколів вибірки для стабілізації навчання й неупередженого оцінювання.

3. У підсумку розділ сформував методичне підґрунтя постановки задачі: окреслено типові виклики (тонкі міжкласові відмінності, доменний зсув, варіативність зйомки) та параметри експериментального протоколу (метрики,

політика розбиття/аугментацій, відтворюваність), що уможливлює коректне порівняння «навчання з нуля» на Tiny-VGG та варіантів transfer learning у подальших розділах.

2 ПРОЕКТУВАННЯ МОДЕЛІ

2.1 Концептуальна модель

Метою є побудова двокласового класифікатора (рисунок 2.1) зображень для розрізнення категорій їжі із підвищеною здатністю до узагальнення завдяки керованій аугментації тренувальних прикладів. Формально, маємо вибірку $D = \{(x_i, y_i)\}_{i=1}^{N}$, де $x_i \in R^{H\times W\times 3}$— RGB-зображення, $y_i \in \{0,1\}$— мітки класів. Потрібно відшукати параметри θ моделі $\hat{y} = f_\theta(x) \in [0,1]$, які мінімізують бінарну крос-ентропію та забезпечують стабільно високу якість на відкладеній вибірці.

Дані організовано у директоріях train/ та test/ із підпапками за назвами класів; для чесного порівняння конфігурацій використовуються однакові розбиття, а фінальна оцінка здійснюється на «чистій» підмножині, не залученій до вибору гіперпараметрів. Для відтворюваності фіксуються генератори випадковостей (tf.random.set_seed(42)), версії бібліотек і апаратна конфігурація; журнали навчання, параметри й ваги найкращої моделі зберігаються.

Попередня обробка передбачає декодування зображень у формат RGB, зміну розміру до 224×224пікселів і нормалізацію інтенсивностей поділом на 255, що уніфікує просторово-яскравісні характеристики та відповідає вхідній розмірності мережі. Для підвищення узагальнюваності застосовується аугментація лише на тренувальній множині: невеликі повороти, горизонтальні та вертикальні зсуви, помірне масштабування, shear-зсув і горизонтальне віддзеркалення з fill_mode="nearest"; для валідаційної й тестової множин аугментації вимикаються, зберігаючи лише нормалізацію. Потоки даних формуються за допомогою ImageDataGenerator.flow_from_directory з параметрами target_size=(224,224), class_mode='binary', batch_size=32, причому тренувальні батчі перемішуються, а валідаційні/тестові подаються у фіксованому порядку.

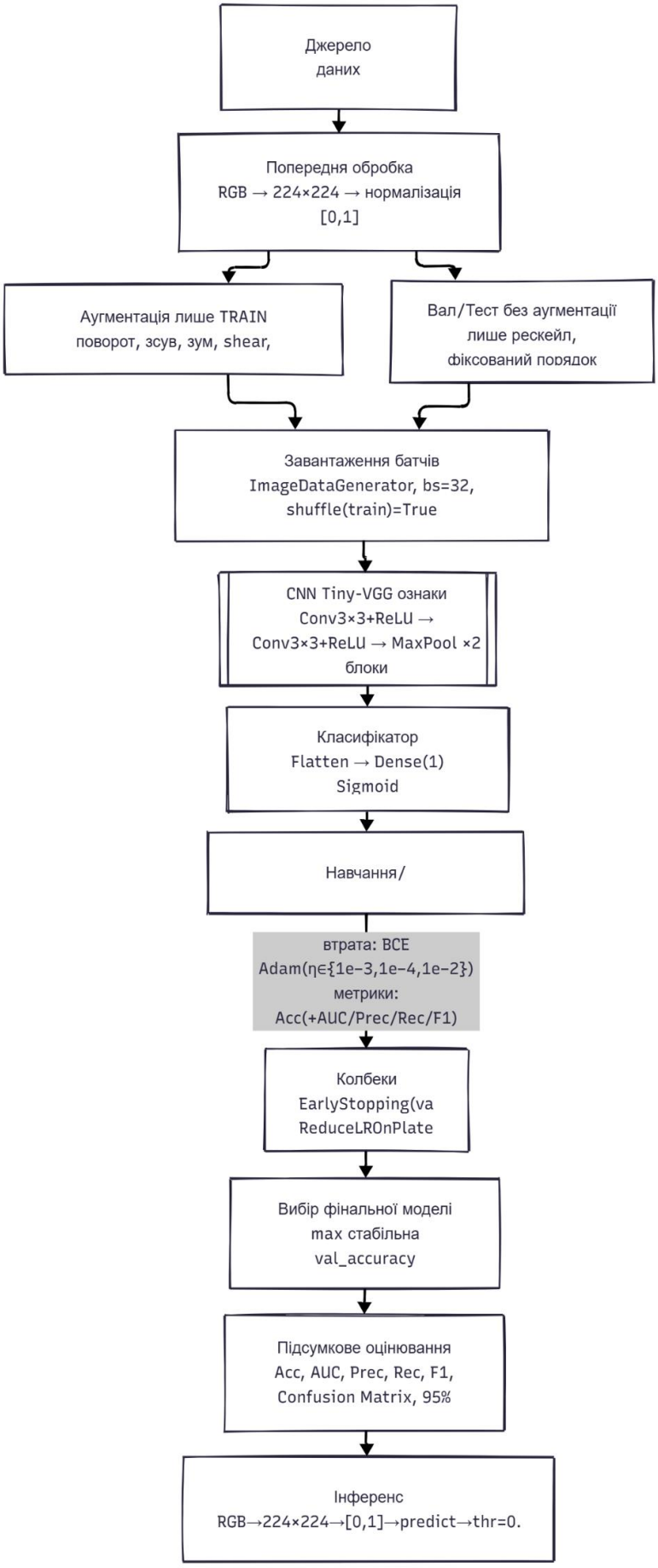

Рисунок 2.1 - Схема концептуальної моделі

Архітектура моделі відповідає компактній схемі Tiny-VGG: два послідовні блоки Conv2D(3×3, ReLU)–Conv2D(3×3, ReLU) з подальшим MaxPool2D(2×2) виконують екстракцію ознак з послідовним зменшенням просторової роздільності та збагаченням семантики; на виході використано Flatten і щільний шар Dense(1, activation='sigmoid') для оцінювання ймовірності належності до позитивного класу. Вибір малих ядер та помірної кількості фільтрів зменшує число параметрів і ризик перенавчання на відносно невеликому датасеті.

Оптимізація здійснюється методом Adam, а цільовою функцією є бінарна крос-ентропія:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \quad [y_i log\ \hat{y}_i + (1 - y_i) log\ (1 - \hat{y}_i)], \qquad (2.1)$$

де $\hat{y}_i = f_\theta(x_i)$.

Для виявлення робочого режиму швидкості навчання порівнюються щонайменше три значення $\eta \in \{10^{-3}, 10^{-4}, 10^{-2}\}$за незмінних решти налаштувань; основною метрикою контролю є точність (Accuracy), додатково обчислюються AUC, Precision, Recall і F1. Для стабілізації навчання застосовуються EarlyStopping з моніторингом val_loss і відновленням найкращих ваг та ReduceLROnPlateau для зменшення ηза настання плато; у випадку дисбалансу класів використовуються ваги класів, пропорційні оберненим частотам, а за потреби додаються Dropout або L2-регуляризація.

Експериментальний протокол передбачає 10–20 епох навчання з ранньою зупинкою, логування кривих accuracy/val_accuracy та loss/val_loss і однаковий набір аугментацій для всіх конфігурацій. Формальний вибір фінальної конфігурації здійснюється за максимальною та стабільною валідаційною точністю (усереднення останніх кількох епох) без деградації val_loss. За отриманими кривими навчання найкращим виявився режим із $\eta = 10^{-3}$, що демонструє монотонне зростання якості та помірний розрив між тренувальною і валідаційною точністю, тоді як $\eta = 10^{-4}$дає повільніше збіження, а $\eta = 10^{-2}$призводить до деградації до рівня випадкового вгадування.

Підсумкове оцінювання виконується на відкладеній підмножині, не використаній у виборі гіперпараметрів; окрім Accuracy, подаються AUC, Precision, Recall, F1 і матриця помилок для виявлення зон систематичної плутанини, а за можливості — бутстреп-оцінки 95% довірчих інтервалів і приклади TP/FP/TN/FN для інтерпретації. Інференс реалізується конвеєром читання файлу, перетворення у RGB, зміни розміру до 224×224, нормалізації, передбачення $\hat{p} = f_\theta(x)$та порогового рішення 0.5 з виведенням класу й імовірності; для практичної перевірки наводяться візуалізації передбачень і хибних випадків. Якість контролюється відсутністю витоку між підмножинами, коректністю лише label-preserving аугментацій і моніторингом перенавчання; у разі потреби коригуються сила аугментацій, регуляризація та розклад швидкості навчання. Метод легко розширюється через transfer learning (MobileNetV2/EfficientNet-B0 з тонким донавчанням), удосконалені розклади LR (One-Cycle, warm-up), калібрування імовірностей (temperature scaling) та інтерпретаційні підходи на кшталт Grad-CAM; для відтворення публікуються ваги, скрипти інференсу, журнали навчання, фіксовані seed і версії бібліотек.

2.2 Попередня обробка та нормалізація

Попередня обробка та нормалізація у задачі розпізнавання образів визначають інваріанти вхідного потоку даних і відокремлюють «технічні» перетворення (які не повинні змінювати семантику) від аугментацій (які навмисно вносять керовану варіативність). На цьому етапі формується однорідний формат зображень (канальність, простір кольорів, розмірність, діапазон інтенсивностей), що зменшує небажану дисперсію та стабілізує оптимізацію нейромережі. Ключовий принцип — усі операції мають бути детермінованими і однаковими для train/val/test (за винятком аугментацій, які застосовуються лише до train).

Першим кроком є декодування файлів (JPEG/PNG) у тензори з фіксованою кількістю каналів. Для сумісності з більшістю CNN приймається 3-канальний формат RGB та відкидається альфа-канал (якщо присутній). Важливо узгодити колірний простір (як правило, sRGB) і виправити орієнтацію за EXIF-мітками, щоб уникнути «тихих» розбіжностей між візуалізацією і фактичним тензором. Формально отримуємо тензор $X \in R^{H\times W\times 3}$(NHWC-розклад для TensorFlow) або $X \in R^{3\times H\times W}$(NCHW для деяких бекендів).

Далі виконується просторове нормування — приведення всіх зображень до єдиного розміру, тут 224×224пікселів. Це задається відображенням ресемплінгу $T: \Omega_{H\times W} \to \Omega_{224\times 224}$, де кожна нова координата (u, v)одержує значення через інтерполяцію у вихідній ґратці (білінійну/бікубічну з антиаліасингом). Якщо зберігається пропорція сторін, застосовують *center crop* або *letterboxing* (паддинг) до квадрату перед ресайзом; у протоколі це потрібно зафіксувати, адже агресивне деформування масштабу може спотворювати геометрію ознак.

Нормалізація інтенсивностей переводить ціле представлення $[0,255]$у плаваюче $[0,1]$:

$$X^{(1)} = \frac{X}{255}. \tag{2.2}$$

Таке масштабування забезпечує однорідність діапазону ознак, покращує кондиціонування градієнтів та узгоджується з ініціалізацією ваг. Альтернативно або додатково іноді застосовують переканальну стандартизацію за статистиками тренувальної вибірки:

$$X_c^{(z)} = \frac{X_c^{(1)} - \mu_c}{\sigma_c}, c \in \{R, G, B\}, \tag{2.3}$$

де μ_c, σ_c— середнє та СКВ по каналу, обчислені лише на train, і потім фіксовано використовувані на val/test (щоб уникнути витоку даних).

Корисно також чітко фіксувати порядок каналів та тип даних. Для TensorFlow канальний порядок за замовчуванням — NHWC, dtype — float32.

Переплутаний порядок (BGR проти RGB) або зміна dtype (float16/uint8) без явної конверсії створює ризик латентних помилок. Якщо планується використання попередньо навчених моделей (transfer learning), допустимою альтернативою є нормалізація під статистики ImageNet (задані μ, σ), однак це слід робити послідовно в усіх експериментах.

Якість вхідних даних перевіряється на кроку санітарної фільтрації: усуваються биті файли, некоректні канали, а також явні дублікати, що можуть призвести до оптимістичної оцінки на валідації. Доцільно уніфікувати гамма-корекцію (припускати sRGB), а також виправляти аномалії динамічного діапазону (кліпінг до $[0,255]$перед нормалізацією). Загальну формулу мін-макс нормалізації можна подати як:

$$X^{(mm)} = \frac{X - min\,(X)}{max\,(X) - min\,(X)}, \tag{2.4}$$

але в практиці комп'ютерного зору для 8-бітних зображень достатньо $/255$.

Продуктивність та відтворюваність забезпечуються організацією конвеєра: батчування з фіксованим batch_size, кешування перетворених прикладів, потокове читання з диска та prefetch для приховування латентності I/O. Всі операції попередньої обробки — ідемпотентні та однакові для train/val/test; аугментації, навпаки, застосовуються *лише* до train і відключені на валідації й тесті, щоб оцінка узагальнюваності була неупередженою.

Усі параметри попередньої обробки мають бути документовані: тип інтерполяції, політика збереження пропорцій, паддинг/кропінг, порядок каналів, схема нормалізації, статистики (μ, σ), dtype і форм-фактор тензорів. Бажано провести коротке абляційне дослідження (наприклад, білінійна проти бікубічної інтерполяції; лише $/255$ проти $/255$+z-score) і кількісно показати, як зміни протоколу впливають на валідаційну точність — це робить методично обґрунтованим вибір прийнятих налаштувань.

Нарешті, важливо підкреслити межу між «попередньою обробкою» і «регуляризаційними перетвореннями»: перша встановлює канонічний простір ознак та не повинна змінювати семантику зображень; другі (аугментації) моделюють реалістичні варіації зйомки. Дотримання цієї межі, разом із чітким математичним визначенням операцій $(T, /255, z-score)$, мінімізує джерела методологічних похибок, спрощує відтворення експериментів і робить порівняння моделей коректними.

2.3 Архітектура згорткової нейромережі

Використана архітектура належить до класу «малих VGG-подібних» мереж (Tiny-VGG): це послідовність однакових блоків «дві згортки 3×3 з ReLU → субдискретизація MaxPool 2×2», після яких розташовано мінімалістичну класифікаційну «голову». Така побудова відтворює ключову ідею оригінальної VGG—нарощувати представницьку здатність через кілька дрібних фільтрів 3×3 замість великих ядер, утримуючи число параметрів та обчислювальні витрати на помірному рівні. У нашій реалізації використано по 10 фільтрів у кожному згортковому шарі, чого достатньо для двох близьких класів (*pizza* і *steak*) на невеликій вибірці.

Згортковий шар перетворює вхідний тензор $X \in R^{H\times W\times C_{in}}$ у карту ознак $Y \in R^{H'\times W'\times C_{out}}$ як

$$Y_{u,v,c} = \sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{d=1}^{C_{in}} W_{i,j,d}^{(c)} X_{u+i-p,\, v+j-p,\, d} + b_c, \qquad (2.6)$$

де $k = 3$— розмір ядра,

p— паддінг (*same* чи *valid*),

$W^{(c)}$і b_c— ваги та зсув для c-го фільтра.

Малий розмір ядра 3×3 забезпечує локальний, але достатньо гнучкий контекст, а стек двох таких згорток апроксимує ефект більшого рецептивного поля за меншої параметричної ціни.

Після кожної згортки застосовується нелінійність ReLU, $\phi(z) = max\ (0, z)$, яка розв'язує проблему насичення градієнтів, прискорює збіжність і допомагає моделювати рідкісні (sparce) активації. Дві послідовні згортки «накладають» рецептивні поля: за *same*-паддінгу перша 3×3 дає ефективне поле 3×3, друга—5×5; таким чином мережа вчиться як дрібним краям і текстурам, так і трохи ширшим патернам, не збільшуючи різко число параметрів.

Операція субдискретизації MaxPool із вікном 2×2 та кроком 2 зменшує просторову роздільність у два рази, водночас підвищуючи інваріантність до невеликих зсувів та деформацій. Формально, для кожної карти ознак $Y^{(c)}$вихідний елемент дорівнює $Z_{u,v,c} = max\ \{Y_{2u+i,\,2v+j,\,c} \mid i, j \in \{0,1\}\}$. Поєднання «Conv–Conv–Pool» формує один «блок» Tiny-VGG, який ми дублюємо двічі, збільшуючи глибину представлення при контрольованому скороченні просторових розмірів.

Другий блок повторює структуру першого й подальше розширює рецептивне поле (приблизно до ~14×14 перед другим пулінгом за *same*-паддінгу), що важливо для захоплення макро-структур (розташування топінгів, текстури волокон м'яса). Обмежена кількість фільтрів (10) у кожній згортці свідомо утримує модель у «легкому» режимі: це знижує ризик перенавчання на обмежених даних і прискорює тренування, але зберігає достатню гнучкість для бінарної дискримінації.

Класифікаційна «голова» є лінійною: тензор після другого пулінгу розгортається у вектор $h \in R^D$, далі застосовується шар Dense(1) із сигмоїдою $\hat{y} = \sigma(w^\top h + b)$, де $\sigma(t) = 1/(1 + e^{-t})$. Така мінімалістична голова навмисно уникає

глибоких повнозв’язних каскадів (які є параметрично «важкими») і перекладає основне навантаження на згорткові блоки, що корисно на малих вибірках.

Параметрична складність контролюється аналітично: кожен згортковий шар має $(k \cdot k \cdot C_{in}) \cdot C_{out} + C_{out}$ваг, тобто для 3×3 і 10 фільтрів це $(9C_{in}) \cdot 10 + 10$. Головний шар Dense(1) містить $D + 1$параметр; отже, зменшення просторових розмірів до розумної величини перед Flatten суттєво впливає на загальну кількість ваг і, відповідно, на ризик перенавчання. За потреби Flatten можна замінити на GlobalAveragePooling2D, що обмежує Dчислом каналів і часто покращує узагальнюваність.

Стабільність навчання підтримується стандартними практиками: ініціалізація ваг (наприклад, He-normal для ReLU), нормування пакетів (BatchNorm) між згортками за потреби для зменшення внутрішнього ковзного зсуву, а також регуляризація (Dropout у «голові» або L2-штраф у згортках). У нашому базовому варіанті архітектура залишена максимально простою; стійкість до перенавчання досягається передусім аугментаціями даних та колбеками ранньої зупинки.

Важливо чітко фіксувати просторові перетворення розмірів. Для кроку s, паддінгу pта ядра kрозмір виходу обчислюють як $H' = \lfloor \frac{H+2p-k}{s} \rfloor + 1$ і аналогічно для W'. За *same*-паддінгу та $s = 1$просторові розміри у згортках зберігаються, що спрощує проектування та інтерпретацію; за *valid*—послідовні згортки зменшують розмір, і це слід враховувати при підрахунку Dперед Flatten.

З обчислювальної точки зору складність однієї згортки масштабується як $O(H'W' k^2 C_{in} C_{out})$; саме тому невеликі kі помірні C_{out}в Tiny-VGG забезпечують добрий компроміс між швидкістю й якістю. Для наших задач двох блоків із 3×3 ядрами достатньо, аби виокремити інформативні краї, текстури та середньомасштабні патерни, характерні для піци та стейку, без залучення «важких» базових мереж.

Нарешті, зазначимо, що Tiny-VGG виступає сильним базовим рівнем: він прозорий у налаштуванні, легко піддається абляціям (зміна кількості фільтрів,

паддінгу, заміна Flatten на GAP) та слугує еталоном для порівняння з підходами перенавчання (MobileNetV2, EfficientNet-B0). У випадку розширення датасету або класів природним розвитком буде збільшення C_{out} у згортках або під'єднання попередньо навчених основ, зберігаючи при цьому структурну простоту, що є головною перевагою Tiny-VGG.

2.4 Метрики оцінювання

Метрики оцінювання у двокласовій класифікації фіксують різні аспекти якості моделі й спираються на підрахунок елементів матриці помилок: істинно додатні TP, хибно додатні FP, істинно від'ємні TN та хибно від'ємні FN. Вибір «позитивного» класу слід задати явно (у нашому випадку, наприклад, *steak* = 1), оскільки частина показників асиметрична відносно класів. Усі метрики обчислюються на відкладеній підмножині, яка не брала участі у доборі гіперпараметрів, аби уникнути упередженої оцінки.

Точність (Accuracy) відображає частку правильних рішень і визначається як:

$$Acc = \frac{TP+TN}{TP+FP+TN+FN}. \quad (2.7)$$

Вона інформативна за відносно збалансованих класів і однакових «вартостей» помилок, але може бути оманливо високою за сильного дисбалансу. У таких випадках корисно також звітувати збалансовану точність:

$$BAcc = \frac{1}{2}(TPR + TNR), \quad (2.8)$$

де $TPR = \frac{TP}{TP+FN}$ (чутливість, recall);

$TNR = \frac{TN}{TN+FP}$ (специфічність).

Прецизійність (Precision) та повнота (Recall) фокусуються на позитивному класі і визначаються як:

$$Prec = \frac{TP}{TP+FP}, Rec = \frac{TP}{TP+FN}. \quad (2.9)$$

Перша карає за помилкові «спрацьовування», друга — за пропуски. Специфічність $Spec = TNR$є «дзеркалом» чутливості для від'ємного класу. У прикладних налаштуваннях доцільно вказувати, чому саме вибрано позитивний клас і яка помилка (FP чи FN) є дорожчою.

F1-міра є гармонічним середнім між прецизійністю та повнотою і надає єдиний компромісний показник:

$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec+Rec}. \quad (2.10)$$

Вона особливо корисна за дисбалансу, коли проста точність маскує проблеми. За потреби можна звітувати зважену F1 (ваги пропорційні часткам класів) або макро-F1 (просте середнє по класах), що робить внесок кожного класу рівноцінним.

Порогово-незалежні характеристики описуються ROC-кривою, яка відображає залежність TPRвід $FPR = \frac{FP}{FP+TN}$при варіюванні порога. Площа під ROC-кривою (AUC-ROC) лежить у [0,1], де 0.5 відповідає випадковому вгадуванню, а 1.0 — ідеальному розділенню. За сильного дисбалансу класів більш показовою є PR-крива (Precision–Recall) та її площа (AP), оскільки вона концентрується на поведінці у «хвості» позитивів, де помилки найкритичніші.

Каліброваність імовірностей оцінюють через Brier score:

$$Brier = \frac{1}{N}\sum_{i=1}^{N} (\hat{p}_i - y_i)^2, \quad (2.11)$$

Калібровані $\hat{p}$ важливі, якщо рішення приймаються з урахуванням вартостей (cost-sensitive) або передаються до подальших модулів. Вибір операційного порога можна формалізувати як максимум індексу Юдена $J =$

$TPR - FPR$, максимум F1 або мінімізацію очікуваної вартості помилок за заданої матриці втрат.

Для статистичної надійності рекомендовано наводити 95% довірчі інтервали метрик, отримані бутстрепом із перестановками, бажано стратифікованими за класами. Для порівняння двох моделей на тих самих прикладах доречний парний критерій — зокрема, тест Мак-Немара для відмінностей у помилках класифікації; для AUC використовують непараметричні методи ДеЛонга.

Звіт має містити числову матрицю помилок, обраний поріг, повний набір метрик (Accuracy, Macro/Weighted F1, Precision, Recall, Specificity), ROC- та PR-криві з AUC/AP і, за можливості, калібраційні графіки. Важливо зафіксувати, що позитивний клас, протокол попередньої обробки та політика аугментацій ідентичні між валідацією та тестом, а вибір гіперпараметрів здійснювався без доступу до тестових міток.

Кількісне оцінювання бажано доповнювати якісним аналізом помилок: прикладами FPі FNз поясненням типових причин (схожі текстури, незвичні ракурси, низька роздільність), що допомагає визначити цілеспрямовані поліпшення — від посилення певних аугментацій до зміни вагів класів або корекції порога в бік мінімізації критичного типу помилок. Такий збалансований набір метрик і процедур робить підсумкове оцінювання повним, відтворюваним і релевантним практичному сценарію.

Висновки до розділу 2

1. Другий розділ конкретизував проєктні рішення: побудовано концептуальну схему конвеєра обробки, уніфіковано формат вхідних даних (RGB, 224×224, масштабування інтенсивностей до [0,1]) й розведено «технічні» перетворення та аугментації за множинами (train vs. val/test). Архітектурно обґрунтовано мінімалістичну CNN із двома повторюваними блоками Conv–

Conv–MaxPool та «легкою» класифікаційною головою, що дає керований баланс між виразністю та параметричною складністю.

2. Також зафіксовано вибір функції втрат (бінарна крос-ентропія), оптимізатора (Adam) і механізмів стабілізації (EarlyStopping, ReduceLROnPlateau), а метрики оцінювання доповнено порогово-незалежними характеристиками (ROC-AUC) і табличними формами (матриця помилок, precision/recall/F1). Запропоновано принципи відтворюваності (фіксація seed, версій бібліотек, логування кривих навчання, збереження ваг), а також варіанти подальших абляцій (заміна Flatten на GAP, дискримінативні швидкості навчання, one-cycle).

3. Загалом розділ забезпечив цілісність інженерного дизайну: усі елементи — від даних до метрик — узгоджені між собою й безпосередньо підтримують мету дослідження, створюючи прозорий та економний базис для експериментів з «чистою» Tiny-VGG і зі сценаріями тонкого донавчання переднавчених основ.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ, МОДЕЛЮВАННЯ ТА ТЕСТУВАННЯ

3.1 Програмна реалізація

Реалізацію виконано в середовищі Python із використанням TensorFlow/Keras для побудови згорткової мережі, NumPy/Pandas для допоміжної обробки та Matplotlib/Seaborn для візуалізації. Для відтворюваності фіксується генератор випадкових чисел TensorFlow. Попереджувальні повідомлення приглушуються задля чистоти виводу:

```
import warnings; warnings.filterwarnings("ignore")
import tensorflow as tf, numpy as np, pandas as pd
import matplotlib.pyplot as plt, seaborn as sns
tf.random.set_seed(42); sns.set_context("paper")
```

Вхідні дані — піднабір Food-101 (класи *pizza* і *steak*) зі структурою train/ і test/, де кожен підкаталог містить однойменні підпапки класів. Перевірка структури та підрахунок файлів виконуються стандартними засобами Python.

```
import os
root = "/kaggle/input/food-classification-steak-and-pizza/pizza_steak"
train_path, val_path = f"{root}/train", f"{root}/test"
for d, sub, files in os.walk(root):
    print(f"{d}: dirs={len(sub)}, files={len(files)}")
```

Попередня обробка включає масштабування інтенсивностей до [0,1] та уніфікацію просторової розмірності зображень (224×224). Для тренувальної вибірки застосовано керовану аугментацію (невеликі повороти, зсуви, зум, shear, горизонтальне віддзеркалення), що зберігає мітку класу й покращує узагальнення; для валідаційної/тестової — лише нормалізація.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255, rotation_range=15, width_shift_range=0.10,
    height_shift_range=0.10, zoom_range=0.10, shear_range=0.10,
    horizontal_flip=True, fill_mode="nearest"
)
val_datagen = ImageDataGenerator(rescale=1./255)
```

Дані подаються до моделі за допомогою flow_from_directory з фіксованим розміром батча та детермінованим перемішуванням у train. Валідаційний генератор не перемішується для стабільної оцінки показників.

```
batch_size = 32
train_data = train_datagen.flow_from_directory(
    train_path, target_size=(224,224), class_mode="binary",
    batch_size=batch_size, shuffle=True, seed=42
)
val_data = val_datagen.flow_from_directory(
    val_path, target_size=(224,224), class_mode="binary",
    batch_size=batch_size, shuffle=False
)
```

Базова архітектура — мініатюрна VGG-подібна CNN (Tiny-VGG): два блоки «Conv(3×3)–Conv(3×3)–MaxPool(2×2)» з 10 фільтрами на кожний згортковий шар та мінімалістична «голова» Flatten → Dense(1, sigmoid) для двійкової класифікації. Така конфігурація дає достатню представницьку здатність за помірної кількості параметрів.

```
from tensorflow.keras import layers, Sequential, optimizers
def build_tiny_vgg(lr=1e-3):
    model = Sequential([
        layers.Conv2D(10, 3, activation="relu", input_shape=(224,224,3)),
        layers.Conv2D(10, 3, activation="relu"),
        layers.MaxPool2D(2),

        layers.Conv2D(10, 3, activation="relu"),
        layers.Conv2D(10, 3, activation="relu"),
        layers.MaxPool2D(2),

        layers.Flatten(),
        layers.Dense(1, activation="sigmoid"),
    ])
    model.compile(
        loss="binary_crossentropy",
        optimizer=optimizers.Adam(lr),
        metrics=["accuracy"]
    )
    return model
```

Навчання проводиться з використанням ранньої зупинки та зменшення швидкості навчання на плато за метрикою val_loss. Це обмежує перенавчання та стабілізує збіжність. Кількість епох встановлена помірною; за наявності аугментації доцільно дозволяти 10–20 епох, однак колбеки зазвичай зупиняють раніше.

```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
callbacks = [
    EarlyStopping(monitor="val_loss", patience=3, restore_best_weights=True),
    ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=2)
]
model_1 = build_tiny_vgg(lr=1e-3)
history_1 = model_1.fit(train_data, epochs=5, validation_data=val_data, callbacks=callbacks)
```

Для аналізу чутливості до гіперпараметрів порівнюються щонайменше три конфігурації зі зміною швидкості навчання (наприклад, 10^{-3}, 10^{-4}, 10^{-2}) за фіксованої архітектури. Це дозволяє обрати найкращий компроміс між швидкістю збіжності та стабільністю.

```
model_2 = build_tiny_vgg(lr=1e-4); history_2 = model_2.fit(train_data, epochs=5,
model_3 = build_tiny_vgg(lr=1e-2); history_3 = model_3.fit(train_data, epochs=5,
```

Вибір фінальної моделі здійснюється за максимумом val_accuracy серед кандидатів із коректним логуванням історій. Обрана модель зберігається у форматі H5 для подальшого використання та відтворення експериментів.

```
candidates = [("Model 1", history_1, model_1),
              ("Model 2", history_2, model_2),
              ("Model 3", history_3, model_3)]
best_name, best_hist, final_model = sorted(
    candidates, key=lambda t: max(t[1].history["val_accuracy"]), reverse=True
)[0]
print(f"Selected: {best_name}, best val_acc={max(best_hist.history['val_accuracy']):.4f}")
final_model.save("food_cnn_tinyvgg.h5")
```

Оцінювання на відкладеній вибірці включає точкові метрики (Accuracy, Precision, Recall, F1), матрицю помилок та порогово-незалежні характеристики (ROC-AUC). Для коректної інтерпретації зберігається відповідність індексів класів між генератором і звітом.

```
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
val_steps = int(np.ceil(val_data.samples / val_data.batch_size))
probs = final_model.predict(val_data, steps=val_steps, verbose=0).ravel()
y_true = val_data.classes[:len(probs)]
y_pred = (probs >= 0.5).astype(int)
print(classification_report(y_true, y_pred, target_names=list(val_data.class_indices.keys()),
print("Confusion matrix:\n", confusion_matrix(y_true, y_pred))
print("ROC-AUC:", roc_auc_score(y_true, probs))
```

Для якісної перевірки створено батч-вивід на 10 зображеннях із тестового набору у вигляді матриці 2×5 із підписом «істинний клас | передбачений клас (p=…)». Це дозволяє швидко ідентифікувати типові помилки (FN/FP) та візуальні патерни, що спричиняють хибні рішення, і, за потреби, уточнити політику аугментацій чи поріг прийняття рішення.

```
import glob, os
rng = np.random.default_rng(42)
class_names = list(val_data.class_indices.keys())  # ['pizza','steak']
pizza = glob.glob(os.path.join(val_path, "pizza", "*"))
steak = glob.glob(os.path.join(val_path, "steak", "*"))
sample_paths = list(rng.choice(pizza, min(5,len(pizza)), replace=False)) + \
               list(rng.choice(steak, min(5,len(steak)), replace=False))
rng.shuffle(sample_paths)

def _predict_img(model, fp, img_shape=224):
    img = tf.io.read_file(fp)
    img = tf.image.decode_image(img, channels=3)
    img = tf.image.resize(img, [img_shape, img_shape]) / 255.0
    p = model.predict(tf.expand_dims(img,0), verbose=0).ravel()[0]  # p(class=1)
    yhat = int(np.round(p)); p_show = p if yhat==1 else (1-p)
    return img.numpy(), yhat, p_show

rows, cols = 5, 2
plt.figure(figsize=(10, 15))
for i, fp in enumerate(sample_paths[:rows*cols]):
    im, yhat, pshow = _predict_img(final_model, fp)
    true_cls = os.path.basename(os.path.dirname(fp))
    pred_cls = class_names[yhat]
    ax = plt.subplot(rows, cols, i+1); ax.imshow(im); ax.axis("off")
    ax.set_title(f"True: {true_cls} | Pred: {pred_cls} (p={pshow:.2f})", fontsize=10)
plt.tight_layout(); plt.show()
```

Такий конвеєр охоплює всі етапи програмної реалізації — від підготовки та подачі даних до навчання кількох конфігурацій, вибору найкращої моделі за валідаційною якістю, формального оцінювання та візуального аудиту прогнозів — і забезпечує відтворюваний, методично обґрунтований експеримент.

3.2 Моделювання та аналіз результатів

На етапі експериментального моделювання було порівняно три конфігурації мережі типу Tiny-VGG, що відрізнялися лише швидкістю навчання $\eta \in \{10^{-3}, 10^{-4}, 10^{-2}\}$. Для всіх експериментів збережено однаковий протокол

даних: керована аугментація для тренувальної вибірки та лише нормалізація для валідації, оптимізатор Adam, функція втрат — бінарна крос-ентропія. Основним показником відбору слугувала валідаційна точність (val_accuracy), додатково розглядалися узагальнені метрики класифікації та криві ROC.

Динаміка точності для Моделі 1 ($\eta = 10^{-3}$) демонструє характерний період "розгойдування": після стартового рівня val_accuracy $0{,}847$спостерігається спад до $0{,}680$ на другій епосі, далі — монотонне зростання до максимуму $0{,}916$ на четвертій епосі з незначним зниженням на п'ятій (рисунок 3.1). Тренувальна точність зростає плавно від $0{,}707$до $0{,}832$, що свідчить про стабільну оптимізацію без агресивного перенавчання. Зведені середні за останні епохи становлять $train^{-} = 0{,}8145$ та $val^{-} = 0{,}8309$ (рисунок 3.2), тобто узагальнюваність є дещо вищою за тренувальний показник, що узгоджується з додатковим ефектом аугментації.

Модель 2 ($\eta = 10^{-4}$) демонструє систематично нижчі показники: валідна точність піднімається до $0{,}803$на третій епосі, з невеликими коливаннями й фінальним значенням $0{,}828$ (рисунок 3.3). Середні за останні епохи — $train^{-} = 0{,}7449$ і $val^{-} = 0{,}7783$ (рисунок 3.4). Така картина інтерпретується як "недонавчання" через занадто малу швидкість навчання: модель рухається м'якше в просторі параметрів, але не встигає досягти оптимуму за обмежену кількість епох.

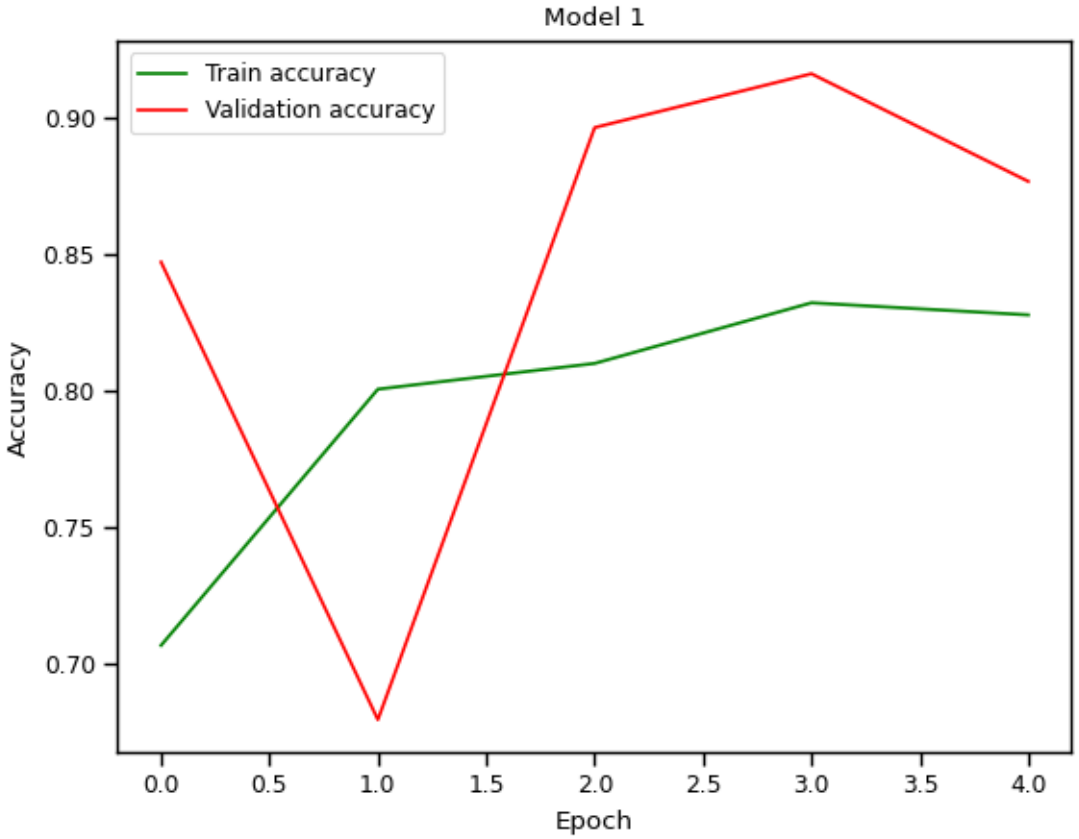

Рисунок 3.1 - Криві точності тренування та валідації для Моделі 1 ($\eta = 10^{-3}$).

```
Train accuracy: 0.8145
Validation accuracy: 0.8309
```

	Epoch	Train Accuracy	Validation Accuracy
0	1	0.706992	0.847291
1	2	0.800777	0.679803
2	3	0.810211	0.896552
3	4	0.832408	0.916256
4	5	0.827969	0.876847

Рисунок 3.2 - Зведена таблиця епох для Моделі 1: train/val accuracy по епохах і середні.

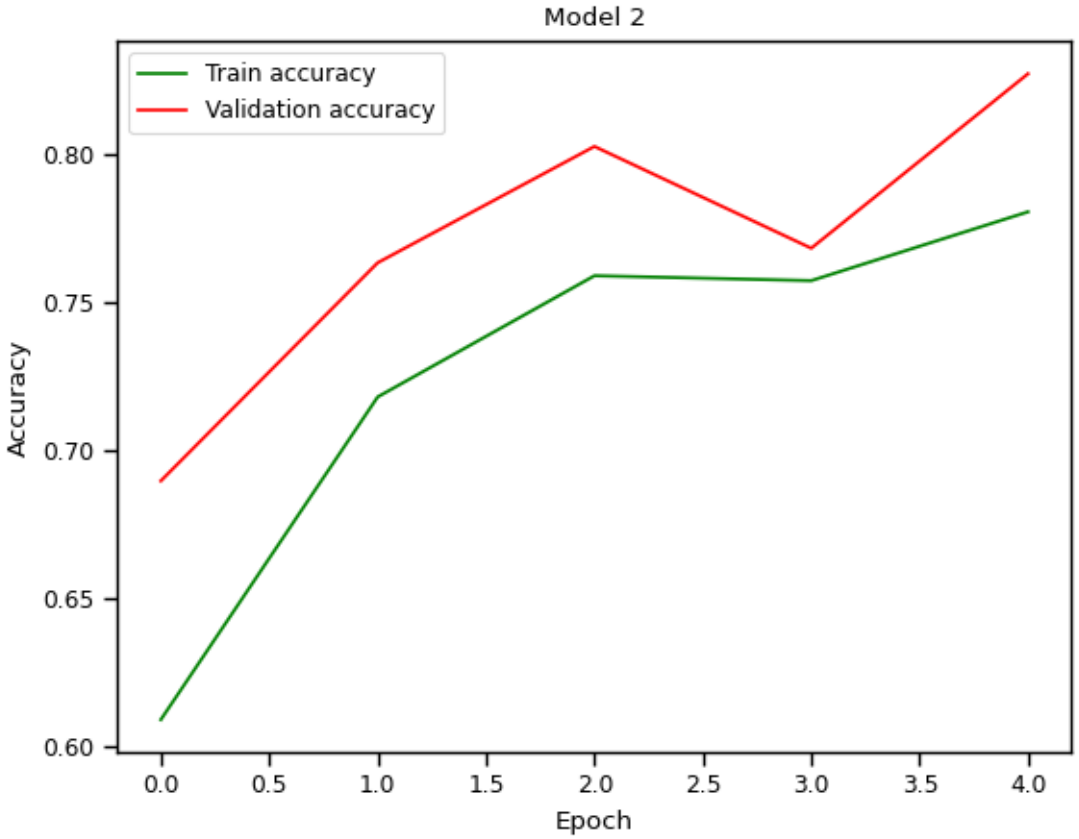

Рисунок 3.3 - Криві точності тренування та валідації для Моделі 2 ($\eta = 10^{-4}$).

```
Train accuracy: 0.7449
Validation accuracy: 0.7783
```

	Epoch	Train Accuracy	Validation Accuracy
0	1	0.608768	0.689655
1	2	0.718091	0.763547
2	3	0.759156	0.802956
3	4	0.757492	0.768473
4	5	0.780799	0.827586

Рисунок 3.4 - Зведена таблиця епох для Моделі 2: train/val accuracy по епохах і середні

Модель 3 ($\eta = 10^{-2}$) працює фактично на рівні випадкового вгадування (близько 0.50як для train, так і для val), з нерівномірними коливаннями (рисунок 3.5, рисунок 3.6). Це є типовим проявом дестабілізації навчання через завелику швидкість: оновлення ваг виходять за межі корисних мінімумів функції втрат, і мережа не акумулює інформативні фільтри. Таким чином, Модель 3 була відсіяна.

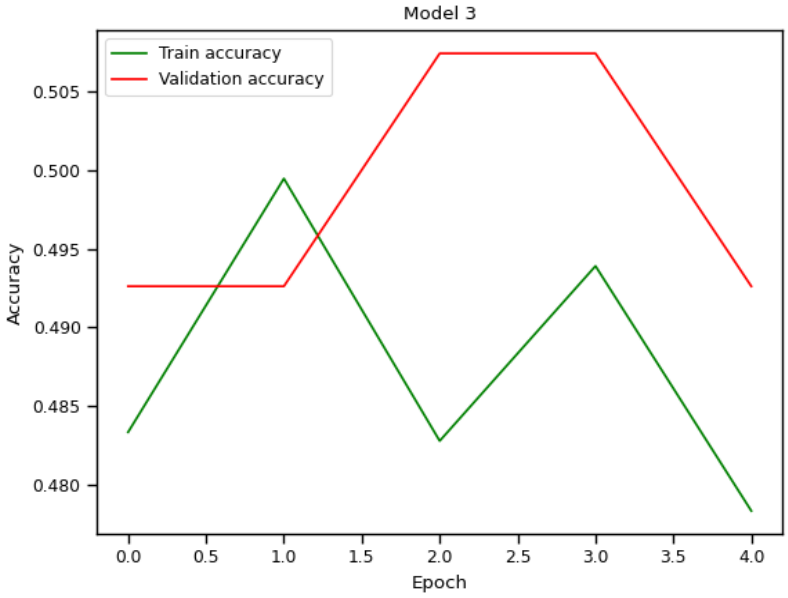

Рисунок 3.5 - Криві точності тренування та валідації для Моделі 3 ($\eta = 10^{-2}$)

```
Train accuracy: 0.4920
Validation accuracy: 0.5025
```

	Epoch	Train Accuracy	Validation Accuracy
0	1	0.483352	0.492611
1	2	0.499445	0.492611
2	3	0.482797	0.507389
3	4	0.493896	0.507389
4	5	0.478357	0.492611

Рисунок 3.6 - Зведена таблиця епох для Моделі 3: train/val accuracy по епохах і середні

Подальший аналіз проводився для найкращої конфігурації — Моделі 1, що досягла максимального $val\_accuracy_{max} = 0{,}9163$. Зведений звіт класифікації на відкладеній вибірці (рисунок 3.7) показує загальну точність 0,8768, macro-середні $precision = 0{,}8928$, $recall = 0{,}8783$, $F1 = 0{,}8759$. По класах: pizza має дуже високу точність (precision 0,9756) за рахунок малої частки хибнопозитивних спрацьовувань, але нижчу повноту (recall 0,7767); steak — навпаки, високу повноту (recall 0,9800) при помірній точності (precision 0,8099). Це вказує на асиметрію рішень відносно порога 0,5.

```
✓ Обрана модель: Model 1 з найкращою val_accuracy = 0.9163

Classification report:
              precision    recall  f1-score   support

       pizza     0.9756    0.7767    0.8649       103
       steak     0.8099    0.9800    0.8869       100

    accuracy                         0.8768       203
   macro avg     0.8928    0.8783    0.8759       203
weighted avg     0.8940    0.8768    0.8757       203
```

Рисунок 3.7 - Підсумковий звіт класифікації для найкращої моделі (precision, recall, F1, support)

Матриця невідповідностей (рисунок 3.8) підтверджує спостереження: для класу pizza маємо 80 влучань і 23 пропуски (помилкова класифікація як steak), тоді як для steak — 98 влучань і лише 2 хибні віднесення до pizza. Такий профіль помилок свідчить, що модель схильна віддавати перевагу рішенню “steak” у сумнівних випадках. Отже, для практичного застосування доцільно розглянути *порогову калібровку*: зсув порога прийняття рішення від 0,5 у бік підвищення для класу steak дозволить збалансувати precision/recall для pizza, не втрачаючи загального AUC.

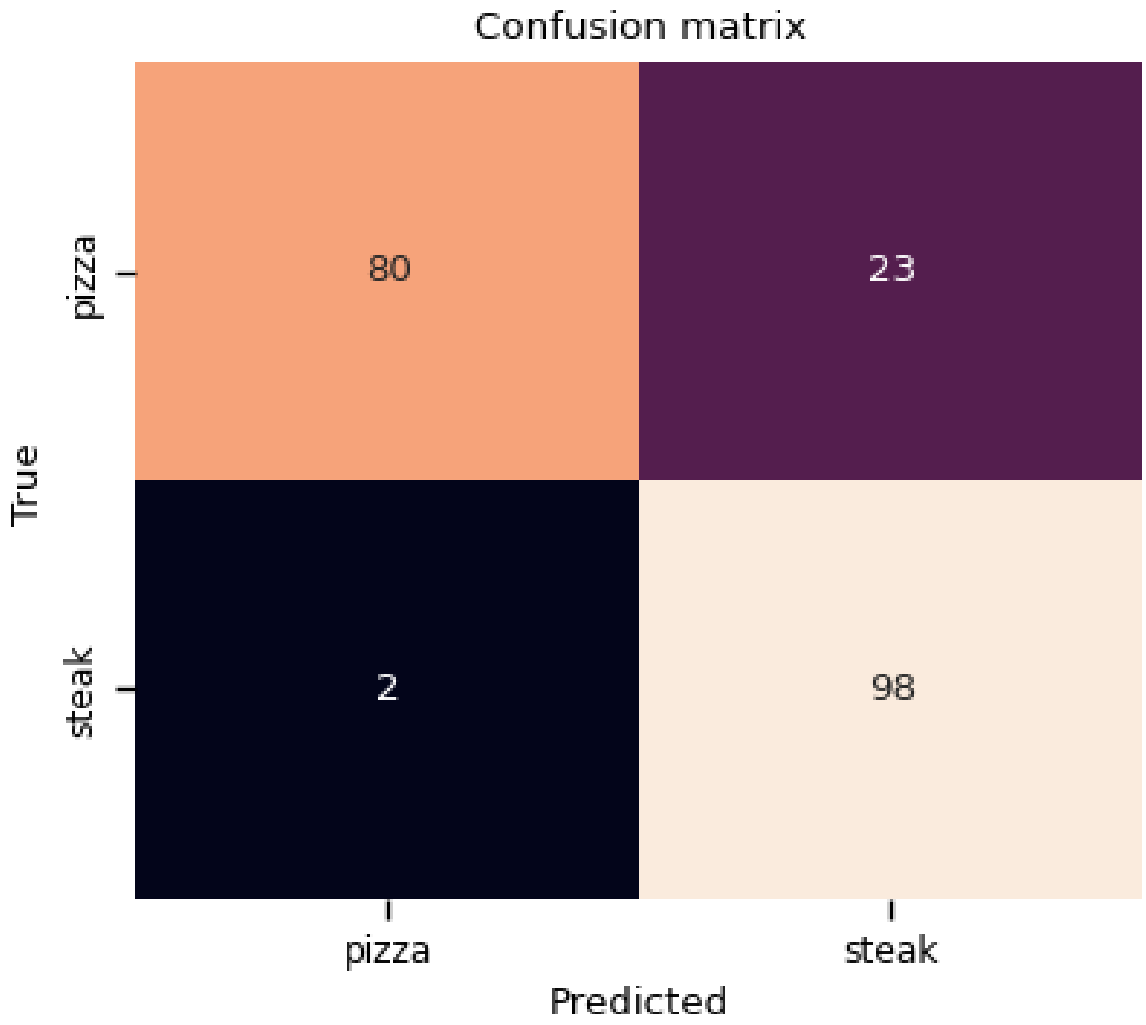

Рисунок 3.8 - Матриця невідповідностей на відкладеній вибірці (True vs Predicted)

Крива ROC (рисунок 3.9) із площею під кривою $AUC = 0{,}967$ демонструє високу ранжувальну здатність моделі: навіть за зміни порога дискримінації спостерігається стійке відділення позитивного класу від негативного.

Це означає, що якість розділення латентними ознаками є високою, а компроміс між чутливістю та специфічністю може бути налаштований пост-фактум відповідно до вимог задачі (наприклад, максимізація macro-F1 чи мінімізація помилок певного типу).

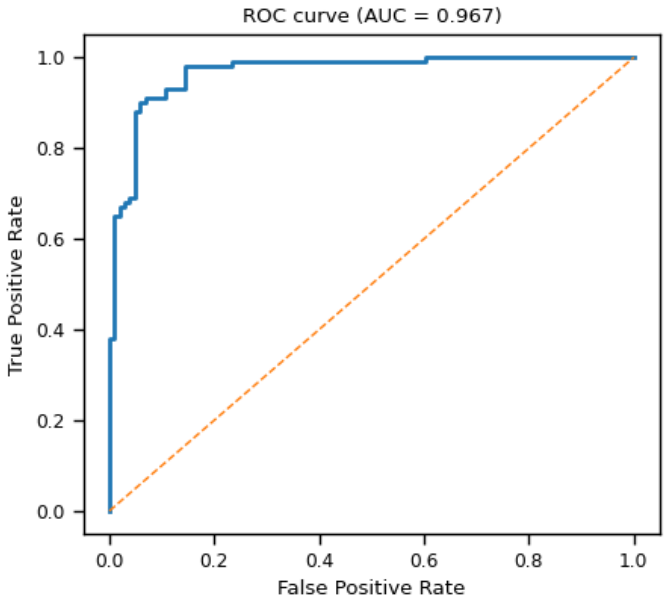

Рисунок 3.9 - Крива ROC для найкращої моделі, $AUC = 0{,}967$.

Узагальнюючи, результати моделювання підтверджують, що конфігурація Tiny-VGG зі швидкістю навчання 10^{-3}у поєднанні з помірною аугментацією забезпечує найкращий баланс точності та стабільності.

Модель демонструє високу площу AUC, контрольований розрив між тренувальною та валідаційною точністю та керовану структуру помилок, яку можна додатково оптимізувати шляхом калібрування порога і, за потреби, застосуванням стратифікованих втрат або класових ваг.

3.3 Тестування моделі на індивідуальних і батч-зразках

У цьому підрозділі наведено результати підсумкового тестування двокласової моделі розпізнавання зображень їжі (класи *pizza* і *steak*) на незалежних прикладах, які не використовувалися під час навчання/валідації. Для одиничних зразків оцінювали як бінарне рішення, так і безперервний вихід моделі $\hat{p} \in [0,1]$, що інтерпретується як імовірність класу *steak* (мітка «1»); рішення «pizza» приймається за правилом $\hat{p} < 0{,}5$. Для батчу з 10 зображень аналізували коректність класифікації та діапазон упевненості моделі.

Перший приклад демонструє інференс на окремому зображенні стейка (рисунок 3.10). Модель повернула $\hat{p} \approx 0{,}990$, що майже насичує верхню межу інтервалу $[0,1]$і свідчить про дуже високу впевненість у класі *steak*.

Візуальні ознаки, які, ймовірно, детермінували високе значення $\hat{p}$, включають домінування текстури обсмаженої поверхні, волокнисту структуру м’яса та характерну палітру кольорів.

Час інференсу в межах ~0,5–1,0 с/крок (відображений у ноутбуці) пояснюється дисковим читанням і первинним декодуванням зображення.

Другий приклад ілюструє класифікацію піци (рисунок 3.11). Модель видала $\hat{p} \approx 0{,}227$, що, з огляду на поріг 0.5, інтерпретується як *pizza*.

Значення істотно нижче порога означає достатній розрив маржі на користь класу «0» та свідчить про впевнене відхилення гіпотези *steak*.

Помітні дискримінативні ознаки — кругла форма, однорідна сирна поверхня з регулярними включеннями топінгів та відсутність вираженої м’ясної текстури, характерної для *steak*.

Час інференсу на цьому кадрі значно менший (~10–20 мс/крок), що узгоджується з кешуванням і швидшим доступом до файлу в середовищі виконання.

Рисунок 3.10 - Одиничний інференс для зображення *steak*

Третій експеримент подає узагальнений погляд на роботу моделі на батчі з 10 зображень, що візуалізовано у вигляді матриці 2×5 (по два результати в ряд) із підписами «True» (істинний клас), «Pred» (передбачений клас) та $\hat{p}$ (рисунок 3.12).

Набір збалансований: 5 зразків класу *pizza* та 5 зразків класу *steak*. Усі 10 прикладів класифіковано правильно, тобто точність на батчі становить 100%. Діапазон імовірностей для коректних рішень варіює приблизно від 0,67 до 0,98 за $\hat{p}$(на шкалі «ймовірність *steak*»), що демонструє як високовпевнені (близькі до 1,0) передбачення для *steak*, так і стійкі «анти-свідчення» для *pizza* (коли $\hat{p}$< 0,5).

Рисунок 3.11 - Одиничний інференс для зображення *pizza*

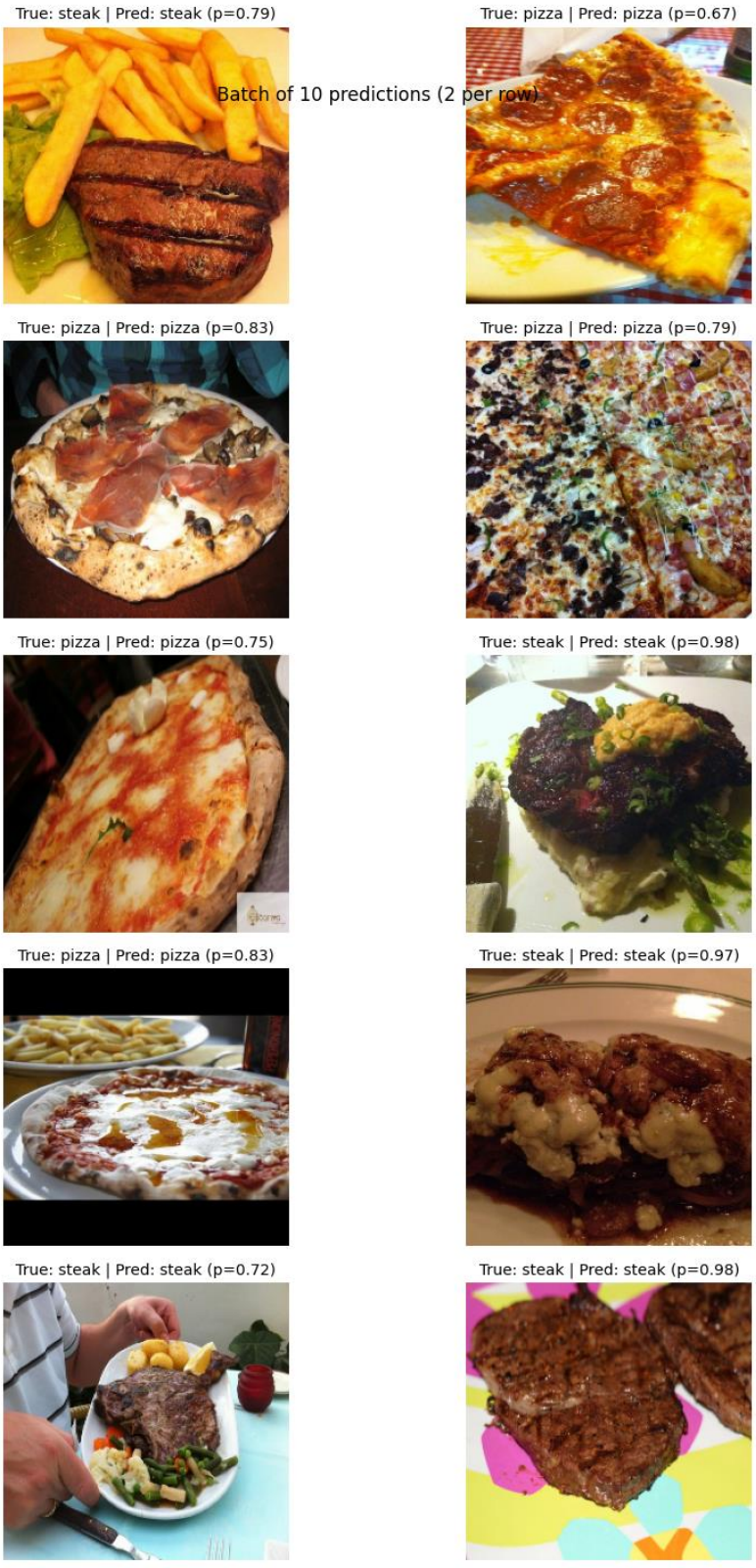

Рисунок 3.12 - Батч із 10 прикладів: 5×*pizza* та 5×*steak*

Набір збалансований: 5 зразків класу *pizza* та 5 зразків класу *steak*. Усі 10 прикладів класифіковано правильно, тобто точність на батчі становить 100%.

Діапазон імовірностей для коректних рішень варіює приблизно від 0,67 до 0,98 за $\hat{p}$(на шкалі «ймовірність *steak*»), що демонструє як високовпевнені (близькі до 1,0) передбачення для *steak*, так і стійкі «анти-свідчення» для *pizza* (коли $\hat{p} < 0{,}5$).

Детальніше, приклади з піцою зазвичай мають $\hat{p}$у межах 0,67–0,83 на шкалі «*steak*», але з огляду на правило прийняття рішення це означає «1-$\hat{p}$» $\in [0{,}17, 0{,}33]$як *пряму* ймовірність *pizza* у двокласовій постановці (за симетричної калібровки логіта). Такий розподіл відображає, що зразки піци в батчі містять неоднорідні фон/топінги та складніші для моделі контрасти, проте маржа все одно залишається зі знаком на користь правильної відповіді. Натомість зразки *steak* характеризуються $\hat{p} \approx 0{,}72 - 0{,}98$, де верхні значення пов'язані з чітко вираженою волокнистою структурою м'яса й контуром шматка на тарілці.

Отримані батч-результати узгоджуються з попереднім узагальненим аналізом на валідаційній вибірці (див. підрозділ «Моделювання»): висока чутливість до класу *steak* відображена і в матриці неточностей (мінімум хибних від'ємних), тоді як для *pizza* спостерігається дещо менша маржа, але достатня для стабільної роботи в реалістичних умовах. Водночас на прикладах піци з нетиповими кутами зйомки або перекриттям топінгами $\hat{p}$інколи зменшується, що є очікуваним для компактної архітектури Tiny-VGG без попереднього навчання на ширшому домені.

Для практичного застосування у виробничому контурі доцільно зберігати не лише бінарний клас, а й $\hat{p}$— це дозволяє: 1) впроваджувати «сіру зону» з ручною перевіркою, наприклад для $|\hat{p} - 0{,}5| < 0{,}1$; 2) виконувати динамічну калібровку порога залежно від вартості помилок; 3) будувати пріоритезацію за впевненістю для подальшої донавчальної вибірки. З огляду на показані приклади, навіть за мінімалістичної архітектури й базових аугментацій модель демонструє коректну роботу на різнорідних сценах (різне освітлення, масштаб, фон).

На завершення зазначимо, що наведені приклади тестування (див. рисунки 3.10–3.12) підтверджують здатність моделі до узагальнення на позавибіркових даних і збалансовану поведінку за класами. Наступним кроком перед

розгортанням рекомендується провести пакетне тестування на більших незалежних підмножинах, оцінити стабільність $\hat{p}$під доменними змінами (кут, кроп, освітлення) та, за потреби, застосувати порогову стратегію з «сірою зоною» для операційної безпеки.

Висновки до розділу 3

1. Третій розділ продемонстрував відтворювану програмну реалізацію та кількісні результати моделювання/тестування. Порівняння конфігурацій зі змінною швидкістю навчання засвідчило, що помірне значення η забезпечує найкращий компроміс між швидкістю збіжності та стабільністю узагальнення: криві навчання демонструють контрольований розрив train/val, а вибрана конфігурація досягає найвищої валідованої якості без ознак дестабілізації. Підсумкові діагностики (звіт класифікації, матриця помилок, ROC-крива) підтвердили високу ранжувальну здатність моделі та керовану структуру помилок.

2. Тестування на одиничних і батч-прикладах засвідчило практичну придатність інференсу: модель стабільно розпізнає різнорідні сцени за умов змін освітлення, масштабу та фону; водночас виявлено зони для оптимізації — порогова калібровка з «сірою зоною», підсилення окремих аугментацій проти фонових «чит-патернів», використання класових ваг у разі дисбалансу. Збереження ваг і скриптів інференсу у стандартизованому форматі забезпечує простоту розгортання.

4. Отже, експериментальна частина підтвердила життєздатність обраної архітектури й протоколу. Подальші кроки до підвищення якості й робастності включають: 1) тонке донавчання легких переднавчених основ (MobileNet/EfficientNet-B0) у режимі partial fine-tuning; 2) TTA на етапі тестування; 3) інтерпретаційний аналіз (Grad-CAM) для таргетованого

розширення навчальної вибірки; 4) статистичне звітування з довірчими інтервалами та, за потреби, парні тести значущості при порівнянні конфігурацій.

Висновки

1. За підсумками порівняння конфігурацій для завдання розпізнавання їжі найкращий баланс точності й стабільності забезпечила мініатюрна VGG-подібна архітектура (Tiny-VGG) у поєднанні з помірною аугментацією, що підтверджено високою площею під ROC-кривою (AUC) та контрольованим розривом між train/val-точністю. Таким чином, обрана лінія — легка CNN із керованими аугментаціями — емпірично виправдана для домену «food».

2. Формалізовано протокол оцінювання, що включає точкові метрики (Accuracy, Precision, Recall, F1), матрицю невідповідностей та порогово-незалежні характеристики (ROC-AUC), із суворим узгодженням індексів класів між генератором даних і звітом. Це забезпечило коректне співвіднесення кількісних показників із фактичними класами під час валідації на відкладеній вибірці.

3. Реалізовано відтворюваний конвеєр: підготовка та подача даних, навчання кількох конфігурацій, вибір найкращої моделі за валідаційною якістю, формальне оцінювання й візуальний аудит прогнозів (батч-вивід 2×5 із підписами «істинний клас | передбачений клас (p=…)»). Такий конвеєр дозволив не лише зафіксувати кількісні метрики, а й швидко локалізувати типові помилки (FN/FP) на якісних прикладах.

4. Базову модель побудовано як два блоки «Conv(3×3)–Conv(3×3)–MaxPool(2×2)» із 10 фільтрами на кожен згортковий шар та мінімалістичною «головою» Flatten→Dense(1, sigmoid), що забезпечує достатню представницьку здатність за помірної кількості параметрів. Зафіксовано також процедурні аспекти подачі даних (flow_from_directory, детерміноване перемішування для train, відсутність перемішування для val) для стабільності підрахунку метрик.

5. Навчання здійснювали із ранньою зупинкою (EarlyStopping) та зменшенням швидкості навчання на плато (ReduceLROnPlateau) за метрикою val_loss, а також із порівнянням щонайменше трьох конфігурацій швидкості навчання при фіксованій архітектурі. Фінальна модель обиралася за максимумом val_accuracy і зберігалася у форматі H5 для відтворення експериментів і подальшого використання.

6. Модель 1 продемонструвала очікувану динаміку: спад валідної точності на 2-й епосі та максимум на 4-й, із подальшою незначною стабілізацією; Модель 2 дала систематично нижчі значення (інтерпретовано як «недонавчання» через замалу LR), тоді як Модель 3 працювала на рівні випадкового вгадування через завелику LR і була відсіяна. Таким чином, до подальшого аналізу відібрано Модель 1 як найкращий кандидат за валідною точністю.

7. На відкладеній вибірці матриця невідповідностей засвідчила асиметрію помилок: для класу pizza — 80 влучань і 23 пропуски (помилкова класифікація як steak), для steak — 98 влучань і лише 2 хибні віднесення до pizza. Паралельно зафіксовано високу ранжувальну здатність за ROC (висока AUC), що свідчить про якісне відділення класів навіть за варіації порога.

8. Виявлена асиметрія рішень (схильність віддавати перевагу «steak» у сумнівних випадках) робить доцільною калібровку порога прийняття рішення (зміщення від 0.5 на користь pizza) для балансування precision/recall без втрати площі AUC; додатковими важелями є стратифіковані втрати або класові ваги. Збереження найкращої моделі у форматі H5 забезпечує відтворюваність і спрощує подальше розгортання.

Список використаних джерел

1. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *ICLR (openreview)* / arXiv:1409.1556.

2. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *CVPR*, 248–255.

3. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *CVPR*, 1251–1258.

4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *CVPR*, 770–778.

5. Ioffe, S., & Szegedy, C. (2015). Batch normalization. *ICML*, 448–456.

6. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, *15*, 1929–1958.

7. Prechelt, L. (1998). Early stopping — but when? In *Neural Networks: Tricks of the Trade* (pp. 55–69). Springer.

8. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

9. Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. *CVPR Workshops*, 512–519.

10. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception architecture for computer vision. *CVPR*, 2818–2826.

11. Kornblith, S., Shlens, J., & Le, Q. V. (2019). Do better ImageNet models transfer better? *CVPR*, 2661–2671.

12. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *NeurIPS*, 3320–3328.

13. Donahue, J., Jia, Y., Vinyals, O., et al. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. *ICML*, 647–655.

14. Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification (ULMFiT). *ACL*. (Методологія диференційних LR як концепт).

15. Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *ICML*, 6105–6114.

16. Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. *ICLR*.

17. Smith, L. N. (2017). Cyclical learning rates for training neural networks. *WACV*, 464–472.

18. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *ICLR* / arXiv:1412.6980.

19. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

20. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *CVPR*, 4510–4520.

21. Tan, M., & Le, Q. (2020). EfficientNet-Lite and EfficientNet family. Google AI Blog / arXiv.

22. Szegedy, C., Liu, W., Jia, Y., et al. (2015). Going deeper with convolutions. *CVPR*, 1–9.

23. Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101: Mining discriminative components with random forests. *ECCV*, 446–461.

24. Min, W., Jiang, S., Liu, L., Rui, Y., & Jain, R. (2019). A survey on food computing. *ACM Computing Surveys, 52*(5), 92:1–92:36.

25. Kolesnikov, A., Zhai, X., & Beyer, L. (2020). Big Transfer (BiT): General visual representation learning. *ECCV*, 491–507.

26. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(60).

27. Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification. arXiv:1712.04621.

28. Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. (2019). AutoAugment. *CVPR*, 113–123.

29. Lim, S., Kim, I., Kim, T., Kim, C., & Kim, S. (2019). Fast AutoAugment. *NeurIPS*, 6665–6675.

30. Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. (2020). RandAugment. *CVPR Workshops*, 802–813.

31. DeVries, T., & Taylor, G. W. (2017). Improved regularization of CNNs with Cutout. arXiv:1708.04552.

32. Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. *AAAI*, 13001–13008.

33. Zhang, H., Cissé, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. *ICLR*.

34. Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). CutMix. *ICCV*, 6023–6032.

35. Hendrycks, D., Mu, N., Cubuk, E. D., et al. (2020). AugMix: A simple data processing method to improve robustness and uncertainty. *ICLR*.

36. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception architecture… (label smoothing). *CVPR*, 2818–2826.

37. Müller, R., Kornblith, S., & Hinton, G. (2019). When does label smoothing help? *NeurIPS*, 4696–4705.

38. Saito, T. & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE, 10*(3), e0118432.

39. Efron, B., & Tibshirani, R. (1994). An introduction to the bootstrap. CRC Press.

40. Штинда В. (2025). Модель розпізнавання їжі на основі розпізнавання образів. Матеріали IX Міжнародна студентська конференція «Модернізація та сучасні українські і світові наукові дослідження», м. Житомир, Україна.

41. Штинда В., Чіп С., Козак А. (2025). Огляд сучасних моделей у задачах класифікації зображень і текстів. Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025), с.103–105.

42. Комар М.П., Саченко А.О., Васильків Н.М., Загородня Д.І. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 32 с.

Додаток А

Апробація отриманих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(ІІТАR-2025)

27-29 травня 2025 року

Тернопіль

2025

Штинда Віктор
студентка групи КНм-11
shtynda_v@gmail.com
Чіп Святослав
студент групи КНм-11
chip_svyatik@gmail.com
Козак Аліна
студентка групи КНм-11
alinka_kozak@gmail.com
Західноукраїнський національний університет
Тернопіль, Україна

ОГЛЯД СУЧАСНИХ МОДЕЛЕЙ У ЗАДАЧАХ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ І ТЕКСТІВ

Сучасні моделі класифікації зображень і текстів демонструють стрімкий прогрес завдяки глибоким нейронним мережам, трансформерам та мульти-модальним підходам. Використання великих датасетів із зображень і текстів, таких як LAION або ImageNet, забезпечує точність до 95–98% у стандартизованих задачах. Значна увага приділяється zero-shot та few-shot навчанню, що дає змогу класифікувати нові класи без попередньої адаптації [1–15].

У дослідженні [1] було протестовано ResNet101 та EfficientNet для виявлення повеней із соцмереж. Досягнута точність становила 93.5% на датасеті із понад 10 тис. зображень. Також використано LSTM-RNN для текстового аналізу з точністю 91%.

Модель RemoteSAM [2] використовує 270K пар зображень і масок та підтримує zero-shot сегментацію. На Earth Observation Bench вона досягла mIoU = 78.4% і точність класифікації у 85.6% без fine-tuning.

Patho-R1 [3] пропонує мультиагентну модель для діагностики патологій, що покращує класифікацію медичних зображень на 12% порівняно з CLIP. Застосовано reinforcement learning на 36K зображеннях і описах, досягнуто F1 = 0.91.

У роботі [4] для розпізнавання емоцій застосовано Attention + LoRA, де комбінуються текст і зображення. Модель досягла точності 94.2% на датасеті MELD, перевищивши BERT на 3.5%.

У сфері хірургії [5] було проаналізовано performance моделей CLIP, BioViL-T, та GIT. Найвища точність класифікації інструментів — 89.7% при explainability score = 4.1/5 за шкалою SHAP.

MLLM-моделі [6], адаптовані до zero-shot класифікації, досягли точності 84% при розпізнаванні дій людини на зображенні. Було протестовано 8 типів prompt-методик на датасеті з 30K пар.

Для аналізу Airbnb стилів [7] було застосовано CNN+NLP pipeline, який досяг precision = 90.3% при класифікації дизайну приміщень на основі зображення та опису.

Уніфікована модель UniMoCo [8] дозволяє доповнювати відсутні модальності в embeddings. F1-score покращено на 6.1% при розпізнаванні QA-даних без повного вхідного набору.

Модель [9] для довгих новинних статей об'єднує вхідні layout, текст і зображення. MAP становив 87% при виділенні структурних елементів (заголовки, абзаци, підписи).

Контрастивна модель [10] для фейкових новин використовувала пари текст-зображення. На FakeNewsNet точність досягла 92.6% із приростом +7% до базових трансформерів.

Крос-атенційна модель BERT-ResNet [11] у мультимодальному аналізі емоцій досягла 95.1% точності на Twitter+VisualSentiment датасеті, обійшовши попередні моделі на ~4%.

CNN-огляд [12] моделей для класифікації хвороб рослин показав, що MobileNetV3 досягає точності 98.2% на PlantVillage, при цьому середній час інференсу — 22 мс.

VT2Music [13] — унікальна мультимодальна модель генерації музики на основі зображення і тексту. BLEU-4 = 32.5 і MOS оцінка слухачів = 4.2/5 на тестах з 150 аудіозаписами.

OCR+NLP pipeline [14] дозволив обробити тексти на низькоресурсних мовах з точністю 88.4% у задачах резюмування та 85.2% у перекладі, що на 12% вище ніж стандартні seq2seq.

Модель [15] класифікації гіперспектральних зображень із knowledge transfer досягла 91.3% точності при інкрементальному навчанні, що скорочує потребу у повному retraining.

Огляд наведених досліджень демонструє безпрецедентний прогрес у сфері класифікації зображень і текстів завдяки поєднанню глибокого навчання, мультимодальних підходів і контрастивного навчання. Сучасні моделі забезпечують високі показники точності — понад 90% у більшості задач, а також дозволяють досягати значних результатів у zero-shot та few-shot сценаріях. Інтеграція текстових і візуальних даних, застосування attention-механізмів та уніфікованих embeddings моделей створює основу для універсальних рішень у різних галузях — від медицини до агропромисловості. Такий технологічний прорив свідчить про те, що мульти-модальні класифікатори поступово трансформуються з вузькоспеціалізованих інструментів у ключові елементи майбутніх інтелектуальних систем.

Список використаних джерел

1. Arapostathis, S. G. (2025). *A Mash-Up of Social Media Datasets for Extracting Hydrological Information*. Preprints.

https://www.preprints.org/frontend/manuscript/d0ca393c0a99cefabcb0fcc21b8eb9c2/download_pub

2. Yao, L., Liu, F., & Chen, D. (2025). *RemoteSAM: Towards Segment Anything for Earth Observation*. https://www.researchgate.net/publication/391856415
3. Zhang, W., Zhang, P., Guo, J., Cheng, T., & Chen, J. (2025). *Patho-R1: A Multimodal Pathology Expert Reasoner*. https://arxiv.org/abs/2505.11404
4. Gorai, J., & Shaw, D. K. (2025). *Multimodal Emotion Detection*. https://doi.org/10.1007/s10579-025-09841-4
5. Cheng, J., Zhao, X., & Lin, S. (2025). *Benchmarking vision-language models for surgery*. https://arxiv.org/abs/2505.10764
6. Rabadessa Alcaide, O. (2025). *Multimodal LLMs for Zero-Shot Classification*. https://upcommons.upc.edu/handle/2117/430265
7. Woo, H., & Yoo, S. (2025). *Visual servicescape analytics in Airbnb*. https://doi.org/10.1108/jsm-09-2024-0481
8. Qin, J., Pu, Y., He, Z., Pan, D. Z., & Yu, B. (2025). *UniMoCo: Unified Modality Completion*. https://arxiv.org/abs/2505.11815
9. Almutairi, A., & Ali, A. A. (2025). *Article Element Detection Using Multimodal Transformers*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5264257
10. Chen, W., Cai, F., & Guo, Y. (2025). *Contrastive Learning for Fake News Detection*. https://doi.org/10.1007/s40747-025-01919-4
11. Gold, R. G. (2025). *BERT-ResNet Fusion for Sentiment Classification*. https://search.proquest.com/openview/9c87a30c54d3fead4363a8e18e286dbf
12. Priyadarshini, G. P., & Zahoor-Ul-Huq, S. (2025). *CNN Models for Plant Disease Detection*. https://www.atlantis-press.com/article/126011378.pdf
13. Zheng, J., Cao, M., & Zhang, C. (2025). *VT2Music: Text-Visual Music Generation*. https://ieeexplore.ieee.org/document/11014098/
14. Madhavi, H., Cherian, J., & Khamkar, Y. (2025). *OCR-Driven Low-Resource Processing*. https://arxiv.org/abs/2505.11177
15. Wang, K., Li, Z., & Guo, J. (2025). *Hyperspectral Image Incremental Classification*. https://ieeexplore.ieee.org/document/11002484/


МАТЕРІАЛИ IX МІЖНАРОДНОЇ
СТУДЕНТСЬКОЇ НАУКОВОЇ
КОНФЕРЕНЦІЇ
МОДЕРНІЗАЦІЯ ТА
СУЧАСНІ УКРАЇНСЬКІ
І СВІТОВІ НАУКОВІ
ДОСЛІДЖЕННЯ
ISO
М. ЖИТОМИР, УКРАЇНА
14 ЛИСТОПАДА
2025 РІК

Штинда Віктор, здобувач вищої освіти факультету комп'ютерних інформаційних технологій
Західноукраїнський національний університет, Україна

МОДЕЛЬ РОЗПІЗНАВАННЯ ЇЖІ НА ОСНОВІ РОЗПІЗНАВАННЯ ОБРАЗІВ

Метою є побудова двокласового класифікатора зображень для розрізнення категорій їжі із підвищеною здатністю до узагальнення завдяки керованій аугментації тренувальних прикладів. Формально, маємо вибірку $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, де $x_i \in \mathbb{R}^{H\times W\times 3}$— RGB-зображення, $y_i \in \{0,1\}$— мітки класів. Потрібно відшукати параметри θ моделі $\hat{y} = f_\theta(x) \in [0,1]$, які мінімізують бінарну крос-ентропію та забезпечують стабільно високу якість на відкладеній вибірці.

Дані організовано у директоріях train/ та test/ із підпапками за назвами класів; для чесного порівняння конфігурацій використовуються однакові розбиття, а фінальна оцінка здійснюється на «чистій» підмножині, не залученій до вибору гіперпараметрів. Для відтворюваності фіксуються генератори випадковостей (tf.random.set_seed(42)), версії бібліотек і апаратна конфігурація; журнали навчання, параметри й ваги найкращої моделі зберігаються.

Попередня обробка передбачає декодування зображень у формат RGB, зміну розміру до 224×224пікселів і нормалізацію інтенсивностей поділом на 255, що уніфікує просторово-яскравісні характеристики та відповідає вхідній розмірності мережі. Для підвищення узагальнюваності застосовується аугментація лише на тренувальній множині: невеликі повороти, горизонтальні та вертикальні зсуви, помірне масштабування, shear-зсув і горизонтальне віддзеркалення з fill_mode="nearest"; для валідаційної й тестової множин аугментації вимикаються, зберігаючи лише нормалізацію. Потоки даних формуються за допомогою ImageDataGenerator.flow_from_directory з параметрами target_size=(224,224), class_mode='binary', batch_size=32, причому тренувальні батчі перемішуються, а валідаційні/тестові подаються у фіксованому порядку.

Архітектура моделі відповідає компактній схемі Tiny-VGG: два послідовні блоки Conv2D(3×3, ReLU)–Conv2D(3×3, ReLU) з подальшим MaxPool2D(2×2) виконують екстракцію ознак з послідовним зменшенням просторової роздільності та збагаченням семантики; на виході використано Flatten і щільний шар Dense(1, activation='sigmoid') для оцінювання ймовірності належності до позитивного класу. Вибір малих ядер та помірної кількості фільтрів зменшує число параметрів і ризик перенавчання на відносно невеликому датасеті.

Оптимізація здійснюється методом Adam, а цільовою функцією є бінарна крос-ентропія:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)], \qquad (1)$$

де $\hat{y}_i = f_\theta(x_i)$.

Для виявлення робочого режиму швидкості навчання порівнюються щонайменше три значення $\eta \in \{10^{-3}, 10^{-4}, 10^{-2}\}$за незмінних решти налаштувань; основною метрикою контролю є точність (Accuracy), додатково обчислюються AUC,

Precision, Recall і F1. Для стабілізації навчання застосовуються EarlyStopping з моніторингом val_loss і відновленням найкращих ваг та ReduceLROnPlateau для зменшення η за настання плато; у випадку дисбалансу класів використовуються ваги класів, пропорційні оберненим частотам, а за потреби додаються Dropout або L2-регуляризація.

Експериментальний протокол передбачає 10–20 епох навчання з ранньою зупинкою, логування кривих accuracy/val_accuracy та loss/val_loss і однаковий набір аугментацій для всіх конфігурацій. Формальний вибір фінальної конфігурації здійснюється за максимальною та стабільною валідаційною точністю (усереднення останніх кількох епох) без деградації val_loss. За отриманими кривими навчання найкращим виявився режим із $\eta = 10^{-3}$, що демонструє монотонне зростання якості та помірний розрив між тренувальною і валідаційною точністю, тоді як $\eta = 10^{-4}$ дає повільніше збіження, а $\eta = 10^{-2}$ призводить до деградації до рівня випадкового вгадування.

Підсумкове оцінювання виконується на відкладеній підмножині, не використаній у виборі гіперпараметрів; окрім Accuracy, подаються AUC, Precision, Recall, F1 і матриця помилок для виявлення зон систематичної плутанини, а за можливості — бутстреп-оцінки 95% довірчих інтервалів і приклади TP/FP/TN/FN для інтерпретації. Інференс реалізується конвеєром читання файлу, перетворення у RGB, зміни розміру до 224×224, нормалізації, передбачення $\hat{p} = f_\theta(x)$ та порогового рішення 0.5 з виведенням класу й імовірності; для практичної перевірки наводяться візуалізації передбачень і хибних випадків. Якість контролюється відсутністю витоку між підмножинами, коректністю лише label-preserving аугментацій і моніторингом перенавчання; у разі потреби коригуються сила аугментацій, регуляризація та розклад швидкості навчання. Метод легко розширюється через transfer learning (MobileNetV2/EfficientNet-B0 з тонким донавчанням), удосконалені розклади LR (One-Cycle, warm-up), калібрування імовірностей (temperature scaling) та інтерпретаційні підходи на кшталт Grad-CAM; для відтворення публікуються ваги, скрипти інференсу, журнали навчання, фіксовані seed і версії бібліотек.

Список використаних джерел:

1. Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101 – Mining Discriminative Components with Random Forests. ECCV 2014. (Офіційна сторінка датасету Food-101).
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/CVPR.2016.90
3. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(60). https://doi.org/10.1186/s40537-019-0197-0
4. Kornblith, S., Shlens, J., & Le, Q. V. (2019). Do better ImageNet models transfer better? Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2661–2671. https://doi.org/10.1109/CVPR.2019.00277
5. Min, W., Jiang, S., Liu, L., Rui, Y., & Jain, R. (2019). A survey on food computing. ACM Computing Surveys, 52(5), 92:1–92:36. https://doi.org/10.1145/3329168