Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмний засіб аналізу медичних зображень для . діагностики пневмонії

Виконав(ла): студент(ка) 4 курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Дмитрів О.Ю. (прізвище та ініціали)

Керівник (підпис) Никитюк В.В. (прізвище та ініціали)

Нормоконтроль (підпис) Липак Г.І. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Коноваленко І.В. (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
"8" червня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Дмитрів Олег Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи **Програмний засіб аналізу медичних зображень для діагностики пневмонії.**

Керівник роботи **к.т.н., доц. Никитюк Вячеслав Вячеславович**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "*14*" *травня* 2026 року № *4/9-239*

2. Термін подання студентом завершеної роботи 19 червня 2026 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ; 1 Теоретичні основи класифікації; 1.1 Пневмонія як медична проблема та роль рентгенодіагностики; 1.2 Загальна характеристика глибокого навчання та згорткових нейронних мережі; 1.3 Концепція трансферного навчання; 1.4 Архітектура MobileNetV2; 1.5 Архітектура VGG16; 1.6 Архітектура DenseNet121; 2 Опис реалізації алгоритмів; 2.1 Загальна архітектура програмного рішення; 2.2 Технологічний стек та бібліотеки; 2.3 Реалізація модуля попередньої обробки даних; 2.4 Реалізація модуля розвідувального аналізу даних; 2.5 Реалізація генераторів даних для навчання моделей; 2.6 Реалізація функції побудови та навчання моделей трансферного навчання; 2.7 Реалізація модуля оцінювання та порівняння моделей; 3 Опис програмного коду; 3.1 Опис обчислювального середовища виконання; 3.2 Результати виконання модуля попередньої обробки даних; 3.3 Результати виконання модуля розвідувального аналізу даних; 3.4 Результати виконання модуля навчання моделей; 3.5 Результати оцінювання моделей на тестовій вибірці; 4 Безпека життєдіяльності, основи охорони праці; Висновки; Список літератури; Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

Титульний слайд; Актуальність дослідження; Трансферне навчання та архітектури; Архітектура конвеєра; Набір даних та попередня обробка; Аугментація та зразки зображень; Результати навчання; Таблиця метрик на тестовій вибірці; Матриці помилок; Аналіз результатів; Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026 – 27.01.2026	*Виконано*
2.	Підбір джерел по темі роботи	28.01.2026 – 01.04.2026	*Виконано*
3.	Оформлення першого розділу	15.04.2026	*Виконано*
4.	Оформлення другого розділу	20.04.2026	
5.	Оформлення третього розділу	30.04.2026	*Виконано*
6.	Виконання завдання до підрозділу "Безпека		
7.	життєдіяльності, основи охорони праці"	15.05.2026	*Виконано*
8.	Оформлення кваліфікаційної роботи	07.06.2026	*Виконано*
9.	Перевірка на плагіат	07.06.2026	*Виконано*
10.	Нормоконтроль	09.06.2026	*Виконано*
11.	Попередній захист кваліфікаційної роботи	11.06.2026	*Виконано*
12.	Захист кваліфікаційної роботи	22.06.2026	
13.			
14.			
15.			

Студент _____ Дмитрів О.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Програмний засіб аналізу медичних зображень для діагностики пневмонії" // Кваліфікаційна робота освітнього рівня "Бакалавр" // Дмитрів Олег Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-41 // Тернопіль, 2026 // с. – 60, рис. – 7, таблиць – 2, джерел – 17, додатків – 1, сторінок додатків – 19.

Ключові слова: глибоке навчання, згорткові нейронні мережі, трансферне навчання, класифікація зображень, пневмонія, рентгенографія, MobileNetV2, VGG16, DenseNet121, медична діагностика, комп'ютерний зір

Кваліфікаційна робота присвячена розробці та порівняльному аналізу програмної системи автоматичної класифікації пневмонії за цифровими рентгенівськими знімками грудної клітки із застосуванням методів глибокого навчання, а саме – трансферного навчання на основі попередньо натренованих згорткових нейронних мереж. Актуальність теми обумовлена високим навантаженням на лікарів-радіологів, потребою у швидких та об'єктивних інструментах підтримки прийняття діагностичних рішень, а також доступністю великих анотованих наборів медичних зображень, що уможливлює застосування сучасних алгоритмів комп'ютерного зору в клінічній практиці.

Об'єктом дослідження є процес автоматизованої класифікації медичних зображень грудної клітки за двома класами – «норма» та «пневмонія». Предметом дослідження є методи, моделі та програмні засоби глибокого навчання, що застосовуються для розв'язання задачі бінарної класифікації

зображень в умовах незбалансованості класів та обмеженого обсягу навчальних даних.

Метою роботи є розробка програмного конвеєра обробки даних та порівняльне дослідження трьох архітектур згорткових нейронних мереж – MobileNetV2, VGG16 та DenseNet121 – для визначення найбільш придатної моделі класифікації пневмонії за критеріями точності, площі під ROC-кривою (AUC), точності (precision) та повноти (recall), з особливим акцентом на мінімізації кількості пропущених випадків захворювання.

У роботі реалізовано повний цикл розробки рішення: збір та перевірку цілісності даних, видалення дублікатів та пошкоджених файлів, формування метаданих, розвідувальний аналіз даних та виявлення дисбалансу класів, розрахунок вагових коефіцієнтів класів, проєктування стратегії аугментації зображень, побудову трьох моделей трансферного навчання, їх навчання з використанням механізмів дострокової зупинки та збереження найкращих ваг, а також комплексне оцінювання результатів на відкладеній тестовій вибірці.

За результатами експериментів встановлено, що архітектура MobileNetV2 продемонструвала найкращий баланс показників якості (AUC = 0,967; точність = 90,9 %; повнота для класу «пневмонія» = 92,0 %) і була обрана як фінальна модель для подальшого використання. Реалізація виконана мовою Python із використанням бібліотек TensorFlow/Keras, scikit-learn, pandas, NumPy, Pillow, Matplotlib та Seaborn у середовищі Jupyter Notebook.

ANNOTATION

"Software Tool for Medical Image Analysis for Pneumonia Diagnostics" // Qualification work of the educational level "Bachelor" // Dmytriv Oleh // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group CH-41 // Ternopil, 2026 // p. – 60, fig. – 7, tables – 2, references – 17, annexes – 1, pages for annexes – 19.

Keywords: deep learning, convolutional neural networks, transfer learning, image classification, pneumonia, chest radiography, MobileNetV2, VGG16, DenseNet121, medical diagnostics, computer vision

The qualification thesis is devoted to the development and comparative analysis of a software system for the automatic classification of pneumonia from digital chest X-ray images using deep learning methods, specifically transfer learning based on pre-trained convolutional neural networks. The relevance of the topic is determined by the high workload of radiologists, the need for fast and objective decision-support tools for medical diagnosis, and the availability of large annotated medical image datasets, which enable the application of modern computer vision algorithms in clinical practice.

The object of the research is the process of automated classification of chest medical images into two classes: “normal” and “pneumonia”. The subject of the research comprises deep learning methods, models, and software tools applied to the task of binary image classification under conditions of class imbalance and limited training data.

The aim of the thesis is to develop a data processing pipeline and conduct a comparative study of three convolutional neural network architectures – MobileNetV2, VGG16, and DenseNet121– in order to determine the most suitable model for pneumonia classification according to the criteria of accuracy, area under

the ROC curve (AUC), precision, and recall, with a particular emphasis on minimizing the number of missed disease cases.

The work implements a complete solution development cycle, including data collection and integrity verification, removal of duplicate and corrupted files, metadata generation, exploratory data analysis and class imbalance detection, calculation of class weights, design of an image augmentation strategy, construction of three transfer learning models, their training using early stopping and best-weight checkpoint mechanisms, as well as comprehensive evaluation on a held-out test dataset.

The experimental results demonstrated that the MobileNetV2 architecture achieved the best balance of performance metrics (AUC = 0.967; accuracy = 90.9%; recall for the “pneumonia” class = 92.0%) and was selected as the final model for further use. The implementation was carried out in Python using the TensorFlow/Keras, scikit-learn, pandas, NumPy, Pillow, Matplotlib, and Seaborn libraries within the Jupyter Notebook environment.

ЗМІСТ

ВСТУП

Пневмонія залишається однією з провідних причин захворюваності та смертності у світі, особливо серед дітей молодшого віку та людей похилого віку. За даними Всесвітньої організації охорони здоров'я, своєчасна та точна діагностика пневмонії має критичне значення для початку адекватного лікування і зниження ризику тяжких ускладнень. Основним інструментом первинної діагностики пневмонії залишається рентгенографія органів грудної клітки, оскільки цей метод є доступним, швидким і відносно дешевим порівняно з комп'ютерною томографією. Водночас інтерпретація рентгенівських знімків вимагає значного досвіду лікаря-радіолога, а навантаження на діагностичні відділення, особливо в умовах епідемій респіраторних захворювань, часто перевищує можливості персоналу опрацювати всі знімки вчасно та без втоми, що позначається на точності діагностики.

У цих умовах застосування методів штучного інтелекту, зокрема глибокого навчання та комп'ютерного зору, розглядається як перспективний напрям підтримки прийняття рішень у медичній діагностиці. Сучасні згорткові нейронні мережі (Convolutional Neural Networks, CNN) здатні автоматично виділяти діагностично значущі візуальні ознаки на зображеннях і за певних умов досягають показників точності, що наближаються до рівня кваліфікованих фахівців. Особливо ефективним підходом для задач з обмеженою кількістю медичних зображень є трансферне навчання (transfer learning), яке дозволяє використовувати знання, накопичені моделлю під час попереднього навчання на великих загальних наборах даних (наприклад, ImageNet), і адаптувати їх до вузькоспеціалізованої задачі – класифікації рентгенівських знімків грудної клітки за наявністю ознак пневмонії.

Актуальність теми кваліфікаційної роботи зумовлена поєднанням кількох факторів: зростаючою доступністю відкритих анотованих наборів медичних зображень, стрімким розвитком апаратного забезпечення для

навчання нейронних мереж, а також практичною потребою охорони здоров'я в інструментах попереднього скринінгу, здатних зменшити навантаження на лікарів-радіологів і прискорити виявлення пневмонії, зокрема в регіонах з обмеженим доступом до висококваліфікованих фахівців.

Об'єктом дослідження є процес автоматизованої бінарної класифікації цифрових рентгенівських знімків грудної клітки на класи «норма» (NORMAL) та «пневмонія» (PNEUMONIA) засобами глибокого навчання.

Предметом дослідження є методи, алгоритми, архітектури згорткових нейронних мереж та програмні засоби, що використовуються для побудови, навчання та оцінювання моделей трансферного навчання у задачі класифікації медичних зображень в умовах незбалансованості класів.

Метою кваліфікаційної роботи є розробка та програмна реалізація конвеєра обробки даних і порівняльне дослідження трьох архітектур згорткових нейронних мереж (MobileNetV2, VGG16, DenseNet121) для визначення моделі, яка забезпечує найкращий баланс точності, площі під ROC-кривою та повноти виявлення пневмонії, з подальшим обґрунтуванням вибору фінальної моделі для практичного застосування.

Для досягнення поставленої мети у роботі вирішуються такі завдання:

1. Проаналізувати теоретичні основи глибокого навчання, згорткових нейронних мереж та трансферного навчання, а також архітектурні особливості моделей MobileNetV2, VGG16 і DenseNet121.

2. Розробити модуль попередньої обробки даних, що забезпечує перевірку цілісності зображень, виявлення та видалення дублікатів і пошкоджених файлів, формування зведеного файлу метаданих.

3. Провести розвідувальний аналіз даних (Exploratory Data Analysis), виявити та кількісно оцінити дисбаланс класів, розробити та обґрунтувати стратегію аугментації зображень і розрахувати вагові коефіцієнти класів.

4. Реалізувати програмний модуль побудови, компіляції та навчання трьох моделей трансферного навчання з використанням механізмів

дострокової зупинки навчання (Early Stopping) та збереження найкращих контрольних точок (Model Checkpoint).

5. Виконати навчання моделей, проаналізувати динаміку навчання за валідаційною вибіркою та оцінити якість класифікації на відкладеній тестовій вибірці за метриками Accuracy, AUC, Precision і Recall.

6. Порівняти отримані результати, обґрунтувати вибір найкращої архітектури для практичного впровадження та сформулювати висновки щодо ефективності застосованого підходу.

Методи дослідження. У роботі застосовано методи теоретичного аналізу та узагальнення наукової літератури з питань глибокого навчання й комп'ютерного зору; методи статистичного аналізу даних для виявлення дисбалансу класів; методи трансферного навчання та тонкого налаштування (fine-tuning) попередньо натренованих згорткових нейронних мереж; методи аугментації зображень; а також методи кількісного оцінювання якості класифікаційних моделей на основі матриці помилок, точності, повноти та площі під ROC-кривою.

Практичне значення отриманих результатів полягає у створенні працездатного програмного конвеєра, який може бути використаний як основа для систем попереднього скринінгу пневмонії за рентгенівськими знімками, а також як навчальний приклад порівняльного дослідження архітектур згорткових нейронних мереж у задачах медичної діагностики із застосуванням мови програмування Python та бібліотек TensorFlow/Keras і scikit-learn.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел. У першому розділі викладено теоретичні основи глибокого навчання, згорткових нейронних мереж, трансферного навчання, розглянуто архітектури MobileNetV2, VGG16 та DenseNet121, а також методи боротьби з незбалансованістю класів і метрики оцінювання якості класифікації. Другий розділ присвячено опису програмної реалізації алгоритмів: архітектури рішення, технологічного стеку

та конкретних програмних модулів. У третьому розділі описано процес виконання програмного коду, наведено отримані результати на кожному етапі та здійснено їх аналіз і порівняння. У висновках узагальнено основні результати роботи.

1 ТЕОРЕТИЧНІ ОСНОВИ КЛАСИФІКАЦІЇ МЕДИЧНИХ ЗОБРАЖЕНЬ ЗАСОБАМИ ГЛИБОКОГО НАВЧАННЯ

1.1 Пневмонія як медична проблема та роль рентгенодіагностики

Пневмонія – це гостре інфекційне захворювання легеневої тканини, що супроводжується запаленням альвеол і порушенням газообміну. Захворювання може мати бактеріальну, вірусну або грибкову природу, а його перебіг суттєво залежить від віку пацієнта, стану імунної системи та своєчасності встановлення діагнозу. Клінічна діагностика пневмонії традиційно ґрунтується на поєднанні фізикального обстеження, лабораторних аналізів і методів медичної візуалізації, серед яких рентгенографія органів грудної клітки залишається найбільш поширеним та економічно доступним інструментом первинного скринінгу.

На рентгенівському знімку пневмонія проявляється як ділянки підвищеної щільності легеневої тканини (інфільтрати), які лікар-радіолог має відрізнити від інших патологічних і фізіологічних варіацій зображення – рубцевих змін, артефактів положення пацієнта, накладання кісткових структур тощо. Інтерпретація таких знімків є суб'єктивним процесом, що залежить від кваліфікації та досвіду лікаря, а в умовах високого навантаження на діагностичні відділення зростає ризик помилок, пов'язаних із втомою або обмеженим часом на аналіз кожного знімка.

Саме ці обставини обумовлюють інтерес до автоматизованих систем підтримки діагностики, здатних виконувати попередній аналіз великої кількості знімків і виокремлювати випадки з високою ймовірністю патології для пріоритетного розгляду лікарем. Важливо зауважити, що метою подібних систем не є повна заміна фахівця, а надання додаткового інструменту, який підвищує швидкість і узгодженість первинного скринінгу. У межах цієї роботи розглядається набір даних Chest X-Ray Images (Pneumonia), що містить 5863 рентгенівські знімки, розподілені на навчальну, валідаційну та тестову вибірки

із розміткою за двома класами – NORMAL (норма) та PNEUMONIA (пневмонія).

1.2 Загальна характеристика глибокого навчання та згорткових нейронних мережі

Глибоке навчання (deep learning) є підрозділом машинного навчання, що ґрунтується на застосуванні багатошарових штучних нейронних мереж для автоматичного виявлення ієрархічних представлень даних. На відміну від класичних методів машинного навчання, де ознаки об'єкта формуються вручну дослідником, глибокі нейронні мережі здатні самостійно навчатися виокремлювати релевантні ознаки безпосередньо з вихідних даних, що особливо важливо для роботи із зображеннями, де ручне конструювання ознак є вкрай трудомістким і малоефективним.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) є спеціалізованим класом глибоких нейронних мереж, архітектура яких враховує просторову структуру зображень. Основними структурними елементами CNN є згорткові шари (convolutional layers), шари підвибірки (pooling layers) та повнозв'язні шари (fully connected layers). Згортковий шар застосовує до вхідного зображення набір навчальних фільтрів (ядер згортки) невеликого розміру, кожен з яких ковзає по зображенню та обчислює локальні відгуки, формуючи карти ознак (feature maps). Така локальна зв'язність та спільне використання вагових коефіцієнтів (weight sharing) суттєво скорочує кількість навчальних параметрів порівняно з повнозв'язними мережами та забезпечує інваріантність до зсуву об'єктів на зображенні.

Шари підвибірки (найчастіше – максимальна підвибірка, max pooling) виконують просторове зменшення розмірності карт ознак, зберігаючи найбільш значущі активації в межах локального вікна, що підвищує обчислювальну ефективність та стійкість мережі до незначних геометричних спотворень вхідного зображення. Глибина мережі, тобто кількість

послідовних згорткових блоків, визначає здатність моделі вивчати ознаки різного рівня абстракції: перші шари зазвичай реагують на прості елементи зображення (краї, контури, перепади яскравості), тоді як глибші шари формують уявлення про складніші структурні та текстурні закономірності, релевантні для конкретної задачі класифікації.

Навчання згорткової нейронної мережі здійснюється методом зворотного поширення помилки (backpropagation) у поєднанні з градієнтними методами оптимізації, серед яких у сучасній практиці найпоширенішим є алгоритм Adam (Adaptive Moment Estimation), що адаптивно коригує крок навчання для кожного параметра окремо на основі накопичених оцінок першого та другого моментів градієнта. Функцією втрат для задач бінарної класифікації традиційно слугує бінарна кросентропія (binary cross-entropy), яка кількісно оцінює відхилення між прогнозованою ймовірністю належності до класу та фактичною міткою.

1.3 Концепція трансферного навчання

Навчання згорткової нейронної мережі «з нуля» (from scratch) на невеликих наборах даних, типових для медичних застосувань, часто призводить до перенавчання (overfitting) – ситуації, коли модель запам'ятовує особливості навчальної вибірки замість виявлення узагальнюючих закономірностей. Крім того, навчання глибоких архітектур з мільйонами параметрів вимагає значних обчислювальних ресурсів і великих обсягів розмічених даних, що рідко доступно в медичних дослідженнях через високу вартість анотування зображень фахівцями.

Трансферне навчання (transfer learning) є підходом, що дозволяє подолати ці обмеження шляхом перенесення знань, накопичених моделлю під час навчання на великому загальному наборі даних, у нову, споріднену за природою задачу. Найпоширенішою практичною реалізацією трансферного навчання у комп'ютерному зорі є використання згорткових нейронних мереж,

попередньо натренованих на наборі даних ImageNet, що містить понад мільйон зображень, розподілених на тисячу категорій. Початкові шари таких мереж навчаються виявляти універсальні низькорівневі візуальні ознаки (краї, текстури, кольорові градієнти), які залишаються релевантними для широкого спектра задач класифікації зображень, включно з медичною візуалізацією, незважаючи на суттєву відмінність доменів.

У типовому сценарії трансферного навчання, застосованому в цій роботі, ваги згорткової частини попередньо натренованої моделі («базової моделі», base model) заморожуються (base_model.trainable = False), тобто не оновлюються під час навчання на нових даних, а навчанню підлягає лише новий класифікаційний блок, доданий над замороженою основою. Така стратегія дозволяє ефективно використовувати вже сформовані ознакові представлення, суттєво скорочуючи кількість параметрів, які потребують навчання, і кількість необхідних навчальних прикладів, водночас зменшуючи ризик перенавчання порівняно з повним навчанням мережі.

У межах роботи над кожною з трьох базових архітектур надбудовується ідентичний класифікаційний блок, що складається із шару глобальної усередненої підвибірки (Global Average Pooling), який перетворює просторові карти ознак на єдиний вектор фіксованої довжини, та повнозв'язного вихідного шару з одним нейроном і сигмоїдною функцією активації, що формує ймовірність належності зображення до класу PNEUMONIA. Така уніфікована конструкція класифікаційної «надбудови» забезпечує коректність порівняння трьох базових архітектур між собою, оскільки відмінності у фінальних показниках якості зумовлені виключно властивостями самих згорткових основ.

1.4 Архітектура MobileNetV2

MobileNetV2 – це згорткова нейронна мережа, спеціально розроблена для застосувань з обмеженими обчислювальними ресурсами, зокрема

мобільних та вбудованих систем. Основою архітектурної ефективності MobileNetV2 є застосування розділюваних за глибиною згорток (depthwise separable convolutions), які розкладають стандартну операцію згортки на дві послідовні: глибинну згортку (depthwise convolution), що обробляє кожен канал вхідного тензора окремо, та точкову згортку (pointwise convolution) розміром 1×1, що комбінує отримані канали. Таке розкладання суттєво зменшує кількість арифметичних операцій і навчальних параметрів порівняно зі стандартною згорткою без істотної втрати якості розпізнавання.

Ключовою інновацією MobileNetV2 порівняно з першою версією є застосування інвертованих залишкових блоків (inverted residual blocks) із лінійними вузькими місцями (linear bottlenecks). На відміну від класичних залишкових з'єднань ResNet, де канали спочатку звужуються, а потім розширюються, у MobileNetV2 вхідний тензор з невеликою кількістю каналів спочатку розширюється точковою згорткою, обробляється глибинною згорткою з нелінійністю ReLU6, а потім знову звужується точковою згорткою без нелінійної активації на виході (лінійний bottleneck), що, за результатами авторів архітектури, дозволяє зберегти більше інформації при проходженні через вузькі шари мережі. Залишкові з'єднання (skip connections) між блоками з однаковою розмірністю входу й виходу полегшують поширення градієнта під час навчання глибокої мережі.

Завдяки описаним архітектурним рішенням MobileNetV2 при вхідному розмірі зображення 224×224×3 пікселі має відносно невелику кількість параметрів і високу обчислювальну швидкість як на етапі навчання, так і на етапі інференсу, що робить цю архітектуру привабливим базовим варіантом (baseline) для порівняння у задачах, де швидкість обробки та можливість розгортання на ресурсообмежених пристроях є важливими практичними критеріями, зокрема для систем підтримки діагностики, що можуть використовуватися безпосередньо в медичних закладах із обмеженою обчислювальною інфраструктурою.

1.5 Архітектура VGG16

VGG16 – класична згорткова нейронна мережа, представлена дослідницькою групою Visual Geometry Group Оксфордського університету, яка стала однією з найвпливовіших архітектур в історії розвитку комп'ютерного зору завдяки демонстрації того, що систематичне збільшення глибини мережі при використанні виключно невеликих згорткових фільтрів розміром 3×3 дозволяє досягати високої якості розпізнавання зображень. Архітектура VGG16 складається із 13 згорткових шарів, об'єднаних у п'ять послідовних блоків, кожен з яких завершується шаром максимальної підвибірки (max pooling), та трьох повнозв'язних шарів у вихідній частині мережі (у класичній реалізації; у варіанті, що використовується для трансферного навчання, повнозв'язні шари видаляються та замінюються новим класифікаційним блоком).

Принциповою архітектурною особливістю VGG16 є послідовне нарощування кількості карт ознак (з 64 у першому блоці до 512 у завершальних блоках) при одночасному зменшенні просторових розмірів карт ознак внаслідок підвибірки, що реалізує типову для CNN ієрархію «від простих ознак до складних абстракцій». Завдяки використанню виключно фільтрів розміром 3×3 при значній глибині мережі, VGG16 здатна моделювати ефективні рецептивні поля великого розміру при контрольованій кількості параметрів на один шар, хоча загальна кількість параметрів мережі (близько 138 мільйонів у повній конфігурації) є значно більшою порівняно з MobileNetV2 та DenseNet121, що обумовлено великою кількістю нейронів у повнозв'язних шарах класичної версії архітектури.

У контексті трансферного навчання VGG16 розглядається як «класичний» репрезентативний приклад глибокої, але архітектурно простої згорткової мережі без спеціалізованих механізмів зменшення обчислювальної складності чи покращення поширення градієнта (на відміну від залишкових з'єднань у MobileNetV2 чи щільних з'єднань у DenseNet121). Це робить VGG16

корисним орієнтиром для порівняння: модель дозволяє оцінити, наскільких переваг надають сучасніші архітектурні рішення порівняно з відносно простою, хоч і глибокою, послідовною структурою згорткових шарів за умов обмеженого донавчання лише класифікаційної надбудови на заморожених вагах базової мережі.

1.6 Архітектура DenseNet121

DenseNet121 (Densely Connected Convolutional Network) – це архітектура, яка вводить принципово інший підхід до організації зв'язків між шарами мережі порівняно з послідовними (VGG16) або залишковими (MobileNetV2) з'єднаннями. У межах кожного щільного блоку (dense block) кожен шар отримує на вхід конкатенацію (об'єднання по каналах) карт ознак усіх попередніх шарів цього блоку, а не лише виходу безпосередньо попереднього шару. Така щільна зв'язність забезпечує максимальне повторне використання ознак (feature reuse), полегшує поширення градієнта під час навчання глибокої мережі та дозволяє суттєво скоротити кількість необхідних параметрів і карт ознак на кожному шарі (тобто параметр «темпу зростання», growth rate, може бути порівняно невеликим) без втрати репрезентативної здатності мережі.

Між щільними блоками розташовані перехідні шари (transition layers), що включають операцію точкової згортки для зменшення кількості каналів та операцію усередненої підвибірки (average pooling) для зменшення просторової розмірності карт ознак, що дозволяє контролювати обчислювальну складність мережі при збереженні щільної внутрішньоблочної зв'язності. Назва DenseNet121 вказує на загальну кількість шарів із навчальними вагами (згорткових і повнозв'язного) в архітектурі – 121.

Завдяки ефективному використанню ознак та стійкості до проблеми згасання градієнта (vanishing gradient) DenseNet121 за літературними даними демонструє високу якість розпізнавання на завданнях медичної візуалізації, де

релевантні діагностичні ознаки можуть бути присутні на різних рівнях абстракції одночасно (наприклад, локальна текстура легеневої тканини в поєднанні із загальною формою і контрастністю інфільтрату). Саме ця властивість обумовила включення DenseNet121 до порівняльного дослідження поряд з MobileNetV2 та VGG16 як представника «сучасного», параметрично ефективного класу архітектур, орієнтованих, зокрема, на задачі медичної діагностики.

1.7 Проблема незбалансованості класів та методи її подолання

Незбалансованість класів (class imbalance) виникає тоді, коли кількість прикладів одного класу в навчальній вибірці суттєво перевищує кількість прикладів іншого класу. У використаному в роботі наборі даних спостерігається саме така ситуація: у навчальній вибірці кількість зображень класу PNEUMONIA майже втричі перевищує кількість зображень класу NORMAL. За відсутності спеціальних заходів модель, що навчається мінімізувати усереднену по вибірці функцію втрат, схильна «зміщуватися» в бік мажоритарного класу, оскільки правильне передбачення часто зустрічаваного класу дає більший внесок у зменшення сумарної помилки, ніж коректна класифікація рідкісного класу. У медичних застосуваннях це особливо небезпечно, якщо мажоритарним класом є патологія, а мінорним – норма (або навпаки), оскільки модель може демонструвати високу загальну точність (accuracy) при систематичній помилковій класифікації менш представленого класу, що в клінічному контексті може призводити або до хибних діагнозів у здорових пацієнтів, або до пропуску реальних випадків захворювання.

Одним з найпоширеніших і обчислювально економних методів боротьби з незбалансованістю класів без зміни складу навчальної вибірки є застосування вагових коефіцієнтів класів (class weights). Суть методу полягає в модифікації функції втрат таким чином, щоб помилки на прикладах менш

представленого класу штрафувалися із більшою вагою, ніж помилки на прикладах домінуючого класу. Збалансовані ваги традиційно обчислюються як обернено пропорційні до частоти класу у навчальній вибірці, наприклад за формулою $w_c = N / (K \cdot n_c)$, де N – загальна кількість навчальних прикладів, K – кількість класів, а n_c – кількість прикладів класу c. Бібліотека scikit-learn реалізує цей підхід у функції compute_class_weight з параметром 'balanced', що використовується у даній роботі для автоматичного розрахунку коефіцієнтів важливості класів NORMAL та PNEUMONIA на основі їхньої фактичної частоти в навчальній вибірці.

Альтернативними підходами до подолання незбалансованості, які не застосовувалися безпосередньо в цій роботі, але широко обговорюються в науковій літературі, є методи передискретизації (oversampling) меншого класу – зокрема, синтетична генерація нових прикладів методом SMOTE – або недискретизації (undersampling) більшого класу, а також застосування спеціалізованих функцій втрат, таких як Focal Loss, що додатково знижує вагу «легких», уже добре класифікованих прикладів на користь складніших випадків. Вибір методу вагових коефіцієнтів класів у цій роботі обумовлений його простотою інтеграції з механізмом навчання Keras (параметр class_weight у методі fit), відсутністю потреби у фізичній зміні розміру вибірки та збереженням повної інформації, що міститься в оригінальних даних.

1.8 Аугментація зображень

Аугментація даних (data augmentation) – це техніка штучного розширення навчальної вибірки шляхом застосування до вихідних зображень керованих, але випадкових перетворень, які не змінюють семантичного класу зображення, але змінюють його конкретне візуальне представлення. Основна мета аугментації полягає у підвищенні узагальнюючої здатності моделі та зменшенні ризику перенавчання, особливо в умовах обмеженого обсягу навчальних даних, типового для медичних застосувань.

Для задачі класифікації рентгенівських знімків грудної клітки важливо обирати перетворення, що є анатомічно й клінічно коректними, тобто не створюють зображень, які не можуть з'явитися в реальній діагностичній практиці. У цій роботі застосовано такі види аугментації: незначний поворот зображення (rotation_range = 10°), що моделює невелику варіативність нахилу пацієнта або апарату під час зйомки; масштабування (zoom_range = 0,1) для моделювання відмінностей у відстані до апарату чи розмірі грудної клітки пацієнта; зсуви по горизонталі та вертикалі (width_shift_range та height_shift_range по 0,1) для моделювання неточного центрування пацієнта в кадрі; а також варіація яскравості (brightness_range = [0,9, 1,1]) для моделювання відмінностей у налаштуваннях експозиції рентгенівського апарата. Принципово важливим рішенням є відмова від горизонтального віддзеркалення зображення (horizontal_flip = False), оскільки дзеркальне відображення рентгенівського знімка грудної клітки порушує анатомічну відповідність лівої та правої сторони тіла пацієнта і може створювати клінічно некоректні, а отже – потенційно шкідливі для навчання моделі – приклади.

Технічно аугментація реалізована засобами класу ImageDataGenerator бібліотеки Keras, який застосовує задані перетворення «на льоту» (real-time) під час формування кожного навчального пакета (batch) даних, тобто оригінальні файли зображень на диску залишаються незмінними, а кожна епоха навчання потенційно «бачить» дещо інші варіанти того самого зображення. Такий підхід не вимагає додаткового дискового простору для зберігання аугментованих копій і забезпечує практично необмежену варіативність навчальних прикладів при фіксованому обсязі вихідного набору даних.

1.9 Метрики оцінювання якості класифікаційних моделей

Коректне оцінювання якості бінарної класифікаційної моделі вимагає використання комплексу взаємодоповнюючих метрик, оскільки жодна окрема

метрика не дає повного уявлення про поведінку моделі, особливо в умовах незбалансованих класів. Базовим інструментом аналізу результатів класифікації є матриця помилок (confusion matrix), що відображає кількість істинно позитивних (True Positive, TP), істинно негативних (True Negative, TN), хибно позитивних (False Positive, FP) та хибно негативних (False Negative, FN) прогнозів моделі відносно фактичних міток класів.

Точність (accuracy) обчислюється як відношення кількості правильних прогнозів до загальної кількості прикладів: Accuracy = (TP + TN) / (TP + TN + FP + FN). Незважаючи на інтуїтивну зрозумілість, ця метрика може бути оманливою за умов незбалансованих класів, оскільки модель, що завжди прогнозує мажоритарний клас, демонструватиме високу точність, попри відсутність реальної діагностичної цінності.

Точність позитивних прогнозів, або precision, обчислюється як Precision = TP / (TP + FP) і відповідає на питання: «яка частка прогнозів моделі про наявність пневмонії є дійсно правильною».

Повнота, або recall (також відома як чутливість, sensitivity), обчислюється як Recall = TP / (TP + FN) і відповідає на питання: «яку частку фактичних випадків пневмонії модель успішно виявила». У медичній діагностиці саме повнота (чутливість) для класу патології є критично важливою метрикою, оскільки хибнонегативний результат (пропущений діагноз пневмонії) може мати значно тяжчі наслідки для пацієнта, ніж хибнопозитивний результат (необхідність додаткового обстеження для здорової людини), що зумовлює особливу увагу до показника Recall (Pneumonia) у порівняльному аналізі моделей цієї роботи.

Площа під ROC-кривою (Area Under the ROC Curve, AUC) є інтегральною метрикою якості бінарного класифікатора, що оцінює здатність моделі розрізняти класи за всіх можливих порогів класифікації, а не лише за фіксованим порогом 0,5, який використовується для обчислення точності, precision і recall.

ROC-крива (Receiver Operating Characteristic) будується в координатах частки хибнопозитивних прогнозів (False Positive Rate) і частки істинно позитивних прогнозів (True Positive Rate, що дорівнює recall) для всього діапазону порогів від 0 до 1.

Значення AUC, що дорівнює 1,0, відповідає ідеальному класифікатору, тоді як значення 0,5 відповідає класифікатору, що працює не краще за випадкове вгадування. Завдяки незалежності від конкретного порогу класифікації AUC вважається найбільш стійкою до незбалансованості класів узагальненою метрикою якості і використовується в цій роботі як основний критерій порівняння трьох моделей, а також як цільова метрика для механізмів дострокової зупинки навчання та збереження найкращих контрольних точок.

1.10 Висновки до першого розділу

У першому розділі роботи розглянуто теоретичні основи, необхідні для розуміння та коректної інтерпретації подальших розділів. Показано, що автоматизована класифікація рентгенівських знімків грудної клітки на класи «норма» та «пневмонія» є актуальною прикладною задачею комп'ютерного зору, розв'язання якої доцільно здійснювати методами трансферного навчання на основі попередньо натренованих згорткових нейронних мереж, що дозволяє ефективно використовувати обмежений за обсягом набір медичних зображень.

Розглянуто принципи побудови та функціонування трьох обраних для порівняння архітектур – MobileNetV2 (з акцентом на ефективність завдяки розділюваним за глибиною згорткам та інвертованим залишковим блокам), VGG16 (як класичний приклад глибокої послідовної мережі на основі фільтрів 3×3) та DenseNet121 (з акцентом на щільну міжшарову зв'язність і ефективне повторне використання ознак).

Розкрито проблему незбалансованості класів, типову для медичних наборів даних, та обґрунтовано застосування методу вагових коефіцієнтів класів як інструменту її подолання. Описано стратегію аугментації зображень,

що враховує анатомічну специфіку рентгенівських знімків грудної клітки, а також систему метрик (Accuracy, Precision, Recall, AUC), яка використовуватиметься для оцінювання та порівняння якості побудованих моделей у наступних розділах роботи.

2 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМІВ

2.1 Загальна архітектура програмного рішення

Програмна реалізація системи класифікації пневмонії побудована як послідовний конвеєр обробки даних (data pipeline), що складається з трьох логічно завершених і фізично відокремлених модулів, кожен з яких реалізований у вигляді окремого обчислювального блокнота Jupyter Notebook (див. рис. 2.1).

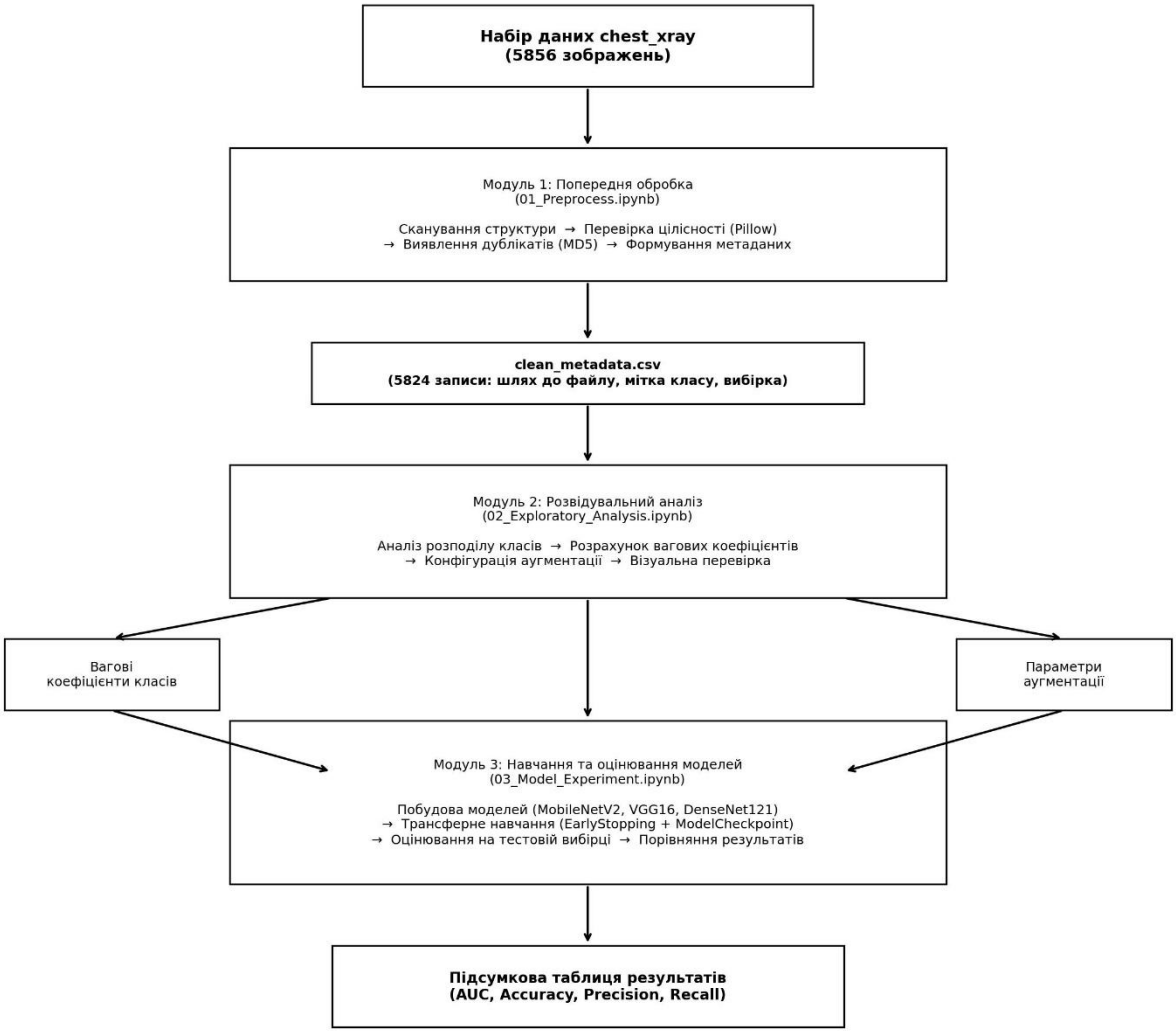


Рисунок 2.1 – Схема архітектури конвеєра опрацювання зображень

Така модульна структура забезпечує відтворюваність результатів, спрощує тестування окремих етапів обробки та дозволяє повторно використовувати проміжні результати без необхідності повторного виконання попередніх, обчислювально витратних кроків.

Перший модуль (01_Preprocess) виконує функцію попередньої обробки: сканування файлової структури набору даних, перевірку цілісності зображень, виявлення та усунення дублікатів і формування єдиного файлу метаданих clean_metadata.csv, який слугує «контрактом» між цим модулем і всіма подальшими етапами обробки. Принциповим архітектурним рішенням є відмова від завантаження пікселів зображень у пам'ять на цьому етапі: метадані містять лише шляхи до файлів, мітки класів та позначення вибірки (train/val/test), що забезпечує високу масштабованість підходу навіть для наборів даних значно більшого обсягу, ніж використаний у роботі.

Другий модуль (02_Exploratory_Analysis) приймає на вхід файл clean_metadata.csv і виконує розвідувальний аналіз даних: побудову статистики розподілу класів за вибірками, розрахунок вагових коефіцієнтів класів, візуалізацію стратегії аугментації та вибіркову візуальну перевірку коректності розмітки. Результати цього модуля мають інформаційний характер і безпосередньо впливають на конфігурацію параметрів навчання моделей у третьому модулі (зокрема, значення вагових коефіцієнтів класів та параметри аугментації).

Третій модуль (03_Model_Experiment) реалізує побудову, навчання та порівняльне оцінювання трьох моделей трансферного навчання на основі архітектур MobileNetV2, VGG16 та DenseNet121. Для забезпечення коректності порівняння всі три моделі навчаються на ідентичних генераторах даних, з однаковими гіперпараметрами навчання (розмір вхідного зображення, розмір пакета, швидкість навчання, кількість епох) та однаковою стратегією колбеків (Early Stopping, Model Checkpoint), що ізолює вплив архітектурних

відмінностей моделей від впливу умов навчання на кінцеві результати порівняння.

2.2 Технологічний стек та бібліотеки

Програмна реалізація виконана мовою Python в інтерактивному середовищі Jupyter Notebook, що забезпечує покрокову розробку, візуалізацію проміжних результатів і документування ходу дослідження безпосередньо поряд із програмним кодом. Для побудови, компіляції, навчання та оцінювання моделей глибокого навчання використано бібліотеку TensorFlow з високорівневим API Keras, що надає готові реалізації попередньо натренованих архітектур (tensorflow.keras.applications), генератори даних із вбудованою підтримкою аугментації (ImageDataGenerator) та механізми колбеків для контролю процесу навчання (EarlyStopping, ModelCheckpoint).

Для роботи з табличними метаданими та їх фільтрації за вибірками й класами використано бібліотеку pandas, для чисельних обчислень (зокрема при розрахунку вагових коефіцієнтів класів) – NumPy. Перевірку цілісності файлів зображень на етапі попередньої обробки виконано засобами бібліотеки Pillow (модуль PIL.Image), яка дозволяє відкрити та перевірити структурну коректність зображення без повного декодування його у масив пікселів, що підвищує швидкість обробки великої кількості файлів. Для обчислення вагових коефіцієнтів класів та формування звітів про якість класифікації (precision, recall, матриця помилок) застосовано бібліотеку scikit-learn (модулі sklearn.utils.class_weight, sklearn.metrics). Візуалізація результатів на всіх етапах роботи виконана засобами бібліотек Matplotlib та Seaborn.

2.3 Реалізація модуля попередньої обробки даних

Реалізація модуля попередньої обробки розпочинається з рекурсивного обходу директорії набору даних, що має фіксовану структуру з трьома

підкаталогами вибірок (train, val, test), кожен з яких містить два підкаталоги класів (NORMAL, PNEUMONIA). Для кожного знайденого файлу зображення формується словник, що містить шлях до файлу, мітку класу та позначення вибірки; усі словники об'єднуються у єдиний об'єкт DataFrame бібліотеки pandas, що надалі служить основною структурою даних для всіх етапів обробки. Програмний код, що реалізує цей етап, наведено в лістингу 2.1.

Лістинг 2.1 – Збір шляхів до файлів зображень та формування початкового набору метаданих

```
# Визначення базової директорії та структури даних
base_dir = "Data/chest_xray"
splits = ["train", "val", "test"]
categories = ["NORMAL", "PNEUMONIA"]

# Формування переліку словників з даними про кожне зображення
image_data = []
for split in splits:
    for category in categories:
        folder_path = os.path.join(base_dir, split, category)
        if not os.path.isdir(folder_path):
            continue
        for img_file in os.listdir(folder_path):
            image_data.append({
                "filepath": os.path.join(folder_path, img_file),
                "label": category,
                "split": split
            })

df = pd.DataFrame(image_data)
print(f"Found {len(df)} total image paths.")
```

Наступним кроком реалізації є перевірка цілісності кожного зображення та виявлення точних дублікатів файлів. Перевірка цілісності здійснюється шляхом спроби відкрити файл засобами Pillow та виклику методу verify(), що перевіряє структурну коректність формату файлу без повного декодування зображення в масив пікселів, що є суттєво швидшим підходом при обробці тисяч файлів. Для виявлення дублікатів для кожного успішно перевіреного файлу обчислюється криптографічний хеш MD5 його бінарного вмісту; якщо обчислений хеш вже зустрічався раніше (зберігається у множині hashes),

відповідний файл вважається точним дублікатом і виключається з подальшої обробки. Такий підхід гарантує виявлення лише абсолютно ідентичних за вмістом файлів (а не візуально подібних, але не тотожних зображень), що є прийнятним компромісом між обчислювальною складністю та повнотою очищення даних для розглянутої задачі. Відповідний програмний код наведено в лістингу 2.2.

Лістинг 2.2 – Перевірка цілісності зображень і виявлення дублікатів за допомогою MD5-хешування

```
def get_file_hash(filepath):
    """Обчислює MD5-хеш файлу."""
    with open(filepath, "rb") as f:
        return hashlib.md5(f.read()).hexdigest()

valid_images = []
hashes = set()
for index, row in tqdm(df.iterrows(), total=df.shape[0]):
    filepath = row['filepath']
    try:
        # 1. Перевірка, що файл є коректним зображенням
        img = Image.open(filepath)
        img.verify()

        # 2. Перевірка на дублікати за допомогою хешування
        file_hash = get_file_hash(filepath)
        if file_hash in hashes:
            continue  # пропустити дублікат

        hashes.add(file_hash)
        valid_images.append(row)

    except Exception as e:
        print(f"Skipping corrupted file: {filepath} due to {e}")

clean_df = pd.DataFrame(valid_images)
clean_df.to_csv("Data/clean_metadata.csv", index=False)
```

Завершальним кроком модуля попередньої обробки є збереження очищеного набору метаданих у файл Data/clean_metadata.csv засобами методу DataFrame.to_csv(). Цей файл стає єдиним інтерфейсом між модулем попередньої обробки та подальшими етапами розвідувального аналізу й навчання моделей, що відповідає принципу розділення відповідальності

(separation of concerns) між модулями конвеєра обробки даних і дозволяє повторно використовувати очищені метадані без необхідності повторного виконання тривалої процедури перевірки цілісності файлів при кожному запуску наступних етапів.

2.4 Реалізація модуля розвідувального аналізу даних

Модуль розвідувального аналізу даних реалізує кілька логічно пов'язаних функцій. Першою є завантаження очищеного файлу метаданих та побудова статистики розподілу зображень за комбінаціями вибірки й класу засобами групування pandas (DataFrame.groupby) у поєднанні з візуалізацією за допомогою функції seaborn.countplot, що дозволяє наочно виявити дисбаланс класів у навчальній вибірці.

Другою функцією є кількісний розрахунок вагових коефіцієнтів класів засобами функції compute_class_weight бібліотеки scikit-learn з параметром стратегії 'balanced', який автоматично обчислює ваги, обернено пропорційні до частоти кожного класу в навчальній вибірці. Розраховані ваги зберігаються у вигляді словника, що відображає числовий індекс класу (0 для NORMAL, 1 для PNEUMONIA) на відповідний ваговий коефіцієнт, і безпосередньо передаються до методу навчання моделі на третьому етапі конвеєра. Реалізацію цього кроку наведено в лістингу 2.3.

Третьою функцією модуля є візуалізація стратегії аугментації зображень: на основі одного зразкового зображення класу PNEUMONIA генерується та відображається п'ять варіантів його аугментованих версій із застосуванням параметрів, обраних з огляду на анатомічну специфіку рентгенівських знімків (див. підрозділ 1.8). Конфігурація генератора аугментації, що використовується як для цієї візуалізації, так і (з додатковим параметром нормалізації пікселів) для подальшого навчання моделей, наведена в лістингу 2.4.

Лістинг 2.3 – Розрахунок збалансованих вагових коефіцієнтів класів

```
from sklearn.utils import class_weight
import numpy as np

train_df = df[df['split'] == 'train']

# Розрахунок збалансованих вагових коефіцієнтів класів
weights = class_weight.compute_class_weight(
    'balanced',
    classes=np.unique(train_df['label']),
    y=train_df['label']
)

class_map = {'NORMAL': 0, 'PNEUMONIA': 1}
class_weight_dict = {
    class_map[label]: weight
    for label, weight in zip(np.unique(train_df['label']), weights)
}
print("Calculated Class Weights:", class_weight_dict)
```

Лістинг 2.4 – Конфігурація генератора аугментації зображень ImageDataGenerator

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Стратегія аугментації, анатомічно коректна для рентгенівських знімків
aug_gen = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    brightness_range=[0.9, 1.1],
    horizontal_flip=False  # анатомічно некоректно для рентгену
)
```

Завершальною функцією модуля розвідувального аналізу є вибіркова візуальна перевірка випадкових зображень кожного класу, що дозволяє переконатися у відповідності шляхів до файлів і присвоєних міток фактичному візуальному змісту зображень, а також якісно оцінити характерні відмінності у вигляді легеневих полів між знімками класів NORMAL і PNEUMONIA.

2.5 Реалізація генераторів даних для навчання моделей

Для забезпечення ефективного навчання моделей на зображеннях, що не завантажуються одночасно в оперативну пам'ять, реалізовано три окремі генератори даних на основі класу ImageDataGenerator та його методу flow_from_dataframe, який дозволяє формувати пакети (batch) зображень безпосередньо з рядків об'єкта DataFrame метаданих за вказаним стовпцем шляху до файлу та стовпцем мітки класу. Навчальний генератор (train_generator) додатково застосовує перетворення аугментації, описані в підрозділі 2.4, та випадкове перемішування порядку прикладів (shuffle=True) для кожної епохи навчання, тоді як валідаційний і тестовий генератори застосовують лише нормалізацію значень пікселів у діапазон [0, 1] (rescale=1./255.) без аугментації та без перемішування порядку прикладів, що забезпечує детермінованість і відтворюваність результатів оцінювання. Усі три генератори налаштовані на уніфікований розмір вхідного зображення 224×224 пікселі (стандартний розмір вхідного шару для всіх трьох порівнюваних архітектур) та розмір пакета 32 приклади. Реалізацію наведено в лістингу 2.5.

2.6 Реалізація функції побудови та навчання моделей трансферного навчання

Центральним елементом програмної реалізації третього модуля є уніфікована функція build_and_train_model, яка приймає на вхід клас базової архітектури Keras (MobileNetV2, VGG16 або DenseNet121) та текстову назву моделі, і виконує повний цикл побудови, компіляції та навчання моделі трансферного навчання. Усередині функції базова модель створюється з вагами, попередньо натренованими на наборі даних ImageNet (weights='imagenet'), без верхнього класифікаційного блоку (include_top=False) та з вхідною формою тензора (224, 224, 3), після чого всі ваги базової моделі

заморожуються (base_model.trainable = False) відповідно до стратегії трансферного навчання, описаної в підрозділі 1.3.

Лістинг 2.5 – Реалізація генераторів даних для навчальної, валідаційної та тестової вибірок

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1./255., rotation_range=10, zoom_range=0.1,
    width_shift_range=0.1, height_shift_range=0.1
)
val_test_datagen = ImageDataGenerator(rescale=1./255.)

train_generator = train_datagen.flow_from_dataframe(
    dataframe=df[df['split'] == 'train'],
    x_col='filepath', y_col='label_idx',
    target_size=IMG_SIZE, class_mode='raw',
    batch_size=BATCH_SIZE, shuffle=True, color_mode="rgb"
)

validation_generator = val_test_datagen.flow_from_dataframe(
    dataframe=df[df['split'] == 'val'],
    x_col='filepath', y_col='label_idx',
    target_size=IMG_SIZE, class_mode='raw',
    batch_size=BATCH_SIZE, shuffle=False, color_mode="rgb"
)

test_generator = val_test_datagen.flow_from_dataframe(
    dataframe=df[df['split'] == 'test'],
    x_col='filepath', y_col='label_idx',
    target_size=IMG_SIZE, class_mode='raw',
    batch_size=BATCH_SIZE, shuffle=False, color_mode="rgb"
)
```

Над замороженою базовою моделлю надбудовується уніфікований класифікаційний блок, що складається з шару глобальної усередненої підвибірки (GlobalAveragePooling2D) та повнозв'язного вихідного шару з одним нейроном і сигмоїдною активацією (Dense(1, activation='sigmoid')), що формує ймовірність належності зображення до класу PNEUMONIA. Модель компілюється з оптимізатором Adam, функцією втрат binary_crossentropy та набором метрик якості, що включає точність (accuracy) і площу під ROC-

кривою (AUC), яка обчислюється під час навчання за допомогою вбудованого класу tf.keras.metrics.AUC.

Процес навчання моделі контролюється двома колбеками Keras. Колбек ModelCheckpoint зберігає на диск (у формат .keras) копію моделі лише тоді, коли значення метрики val_auc на поточній епосі перевищує найкраще зафіксоване раніше значення (save_best_only=True, monitor='val_auc', mode='max'), що гарантує збереження саме найкращої, а не останньої версії моделі, навіть якщо в подальших епохах якість моделі знижується через перенавчання. Колбек EarlyStopping відстежує ту саму метрику val_auc і автоматично зупиняє процес навчання, якщо протягом п'яти послідовних епох (patience=5) не спостерігається покращення, після чого відновлює ваги моделі до найкращого зафіксованого стану (restore_best_weights=True), що дозволяє економити обчислювальні ресурси та запобігати перенавчанню без необхідності заздалегідь точно визначати оптимальну кількість епох навчання. Максимальна кількість епох встановлена на рівні 20, проте фактична тривалість навчання кожної моделі визначається динамічно механізмом дострокової зупинки. Під час навчання моделі також застосовуються попередньо розраховані вагові коефіцієнти класів (class_weight=class_weight_dict), що передаються безпосередньо до методу model.fit(). Повну реалізацію функції побудови й навчання моделі, а також цикл її послідовного застосування до трьох архітектур, наведено в лістингу 2.6.

Лістинг 2.6 – Функція побудови, компіляції та навчання моделі трансферного навчання

```
def build_and_train_model(base_model_class, model_name):
    """Будує, компілює та навчає модель трансферного навчання."""

    # Побудова моделі
    base_model = base_model_class(
        weights='imagenet', include_top=False,
        input_shape=(224, 224, 3)
    )
```

```
    base_model.trainable = False
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=base_model.input, outputs=predictions)

    # Компіляція моделі
    model.compile(
        optimizer=Adam(learning_rate=LEARNING_RATE),
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
    )

    # Колбеки навчання
    checkpoint = ModelCheckpoint(
        f"best_{model_name}.keras",
        save_best_only=True, monitor="val_auc", mode="max"
    )
    early_stopping = EarlyStopping(
        monitor="val_auc", patience=5,
        mode="max", restore_best_weights=True
    )

    # Навчання моделі
    history = model.fit(
        train_generator,
        epochs=EPOCHS,
        validation_data=validation_generator,
        class_weight=class_weight_dict,
        callbacks=[checkpoint, early_stopping],
        verbose=1
    )
    return model, history

models_to_train = {
    "MobileNetV2": MobileNetV2,
    "VGG16": VGG16,
    "DenseNet121": DenseNet121
}

trained_models, histories = {}, {}
for name, model_class in models_to_train.items():
    model, history = build_and_train_model(model_class, name)
    trained_models[name] = model
    histories[name] = history
```

2.7 Реалізація модуля оцінювання та порівняння моделей

Реалізацію циклу оцінювання моделей наведено в лістингу 2.7; детальний аналіз отриманих числових результатів та відповідних графічних матеріалів представлено в розділі 3 цієї роботи.

Лістинг 2.7 – Цикл оцінювання моделей на тестовій вибірці та формування підсумкової таблиці результатів

```
results = []
for name, model in trained_models.items():
    # Завантаження найкращих ваг, збережених ModelCheckpoint
    best_model = tf.keras.models.load_model(f"best_{name}.keras")

    # Обчислення метрик на тестовій вибірці
    loss, acc, auc = best_model.evaluate(test_generator, verbose=0)

    y_pred = (best_model.predict(test_generator) > 0.5).astype(int)
    y_true = test_generator.labels
    report = classification_report(
        y_true, y_pred, target_names=['NORMAL', 'PNEUMONIA'],
        output_dict=True
    )

    results.append({
        "Model": name, "Test AUC": auc, "Test Accuracy": acc,
        "Precision (Pneumonia)": report['PNEUMONIA']['precision'],
        "Recall (Pneumonia)": report['PNEUMONIA']['recall']
    })

    cm = confusion_matrix(y_true, y_pred)

results_df = pd.DataFrame(results).set_index("Model")
```

Після завершення навчання всіх трьох моделей виконується їх оцінювання на відкладеній тестовій вибірці, яка не використовувалася ані для навчання, ані для прийняття рішень про дострокову зупинку навчання, що забезпечує об'єктивність порівняння. Для кожної моделі завантажуються саме найкращі збережені ваги (а не стан моделі на момент завершення навчання) засобами функції tf.keras.models.load_model(), що звертається до файлу, створеного колбеком ModelCheckpoint на етапі навчання.

Для кожної моделі обчислюються чотири ключові показники якості: тестова втрата (loss), тестова точність (accuracy) та площа під ROC-кривою (AUC) – засобами методу model.evaluate(); а також точність (precision) і повнота (recall) для класу PNEUMONIA – засобами функції classification_report бібліотеки scikit-learn, застосованої до бінаризованих прогнозів моделі (з порогом 0,5) та фактичних міток тестової вибірки.

Додатково для кожної моделі обчислюється та візуалізується матриця помилок (confusion_matrix) у вигляді теплової карти (heatmap) засобами бібліотеки Seaborn, що дозволяє наочно проаналізувати розподіл усіх чотирьох типів прогнозів (TP, TN, FP, FN) для кожної з трьох моделей.

Розраховані показники для всіх трьох моделей об'єднуються в єдину підсумкову таблицю результатів (results_df) на основі об'єкта pandas.DataFrame з індексацією за назвою моделі, що дозволяє безпосередньо порівняти моделі за кожним з чотирьох критеріїв та виділити найкращі значення для прийняття обґрунтованого рішення щодо фінальної моделі.

2.8 Висновки до другого розділу

У другому розділі описано програмну реалізацію розробленого рішення, що складається з трьох послідовних модулів – попередньої обробки даних, розвідувального аналізу та побудови, навчання й оцінювання моделей трансферного навчання. Обґрунтовано вибір технологічного стеку на основі мови Python та бібліотек TensorFlow/Keras, pandas, NumPy, Pillow, scikit-learn, Matplotlib і Seaborn, кожна з яких виконує чітко визначену функціональну роль у загальному конвеєрі обробки даних.

Детально розкрито програмну реалізацію кожного етапу: збір метаданих та перевірку цілісності зображень за допомогою MD5-хешування; розрахунок вагових коефіцієнтів класів та конфігурацію стратегії аугментації; уніфіковану функцію побудови й навчання моделей трансферного навчання з механізмами автоматичної дострокової зупинки та збереження найкращих контрольних точок; а також процедуру комплексного оцінювання моделей на тестовій вибірці за чотирма ключовими метриками якості. Архітектурна послідовність та уніфікованість умов навчання й оцінювання всіх трьох моделей забезпечують методологічну коректність їх подальшого порівняння, результати якого детально проаналізовано в розділі 3.

3 ОПИС ВИКОНАННЯ ПРОГРАМНОГО КОДУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Опис обчислювального середовища виконання

Усі етапи розробленого конвеєра обробки даних та навчання моделей було виконано в середовищі Jupyter Notebook з використанням мови програмування Python версії 3.13. Основними бібліотеками виконання були TensorFlow/Keras (для побудови, навчання та оцінювання нейронних мереж), pandas і NumPy (для роботи з табличними та числовими даними), Pillow (для перевірки цілісності зображень), scikit-learn (для розрахунку вагових коефіцієнтів класів та метрик якості класифікації), а також Matplotlib і Seaborn (для побудови графічних матеріалів). Вихідний набір даних складався із 5856 зображень рентгенограм органів грудної клітки, розподілених на три вибірки – навчальну, валідаційну та тестову, кожна з яких містить два класи: NORMAL (норма) та PNEUMONIA (пневмонія). Результати виконання кожного з трьох програмних модулів детально розглянуто в наступних підрозділах.

3.2 Результати виконання модуля попередньої обробки даних

У результаті виконання першого модуля було здійснено повний обхід директорії набору даних та сформовано початковий перелік із 5856 зображень із зазначенням мітки класу та належності до відповідної вибірки. Під час перевірки цілісності файлів засобами Pillow та виявлення точних дублікатів за допомогою MD5-хешування було виявлено й виключено 32 файли – пошкоджені або повторювані зображення, що не несуть додаткової інформаційної цінності для навчання моделі та можуть негативно вплинути на якість оцінювання за наявності ідентичних копій одночасно в різних вибірках. Підсумковий очищений набір метаданих, що містить 5824 зображення, було

збережено у файл Data/clean_metadata.csv. Розподіл кількості зображень за вибірками та класами до та після очищення наведено в таблиці 3.1.

Таблиця 3.1 – Розподіл зображень за вибірками та класами до і після очищення даних

Вибірка	**Клас**	**До очищення**	**Після очищення**
train	NORMAL	1341	1340
train	PNEUMONIA	3875	3850
val	NORMAL	8	8
val	PNEUMONIA	8	8
test	NORMAL	234	231
test	PNEUMONIA	390	387
Усього	—	5856	5824

Як видно з таблиці 3.1, абсолютна більшість виключених файлів (32 з 32) припадала на навчальну та тестову вибірки класу PNEUMONIA, тоді як валідаційна вибірка залишилася без змін. Це підтверджує коректність та обережність застосованої процедури очищення: вона не призвела до втрати жодного зображення з украй малочисельної валідаційної вибірки (16 зображень), яка є критично важливою для контролю процесу навчання моделей через колбек EarlyStopping.

3.3 Результати виконання модуля розвідувального аналізу даних

Аналіз розподілу класів за вибірками, виконаний на основі очищеного набору метаданих, наочно підтвердив наявність суттєвого дисбалансу класів у навчальній вибірці: кількість зображень класу PNEUMONIA (3850) майже втричі перевищує кількість зображень класу NORMAL (1340). Графічне представлення цього розподілу наведено на рисунку 3.1.

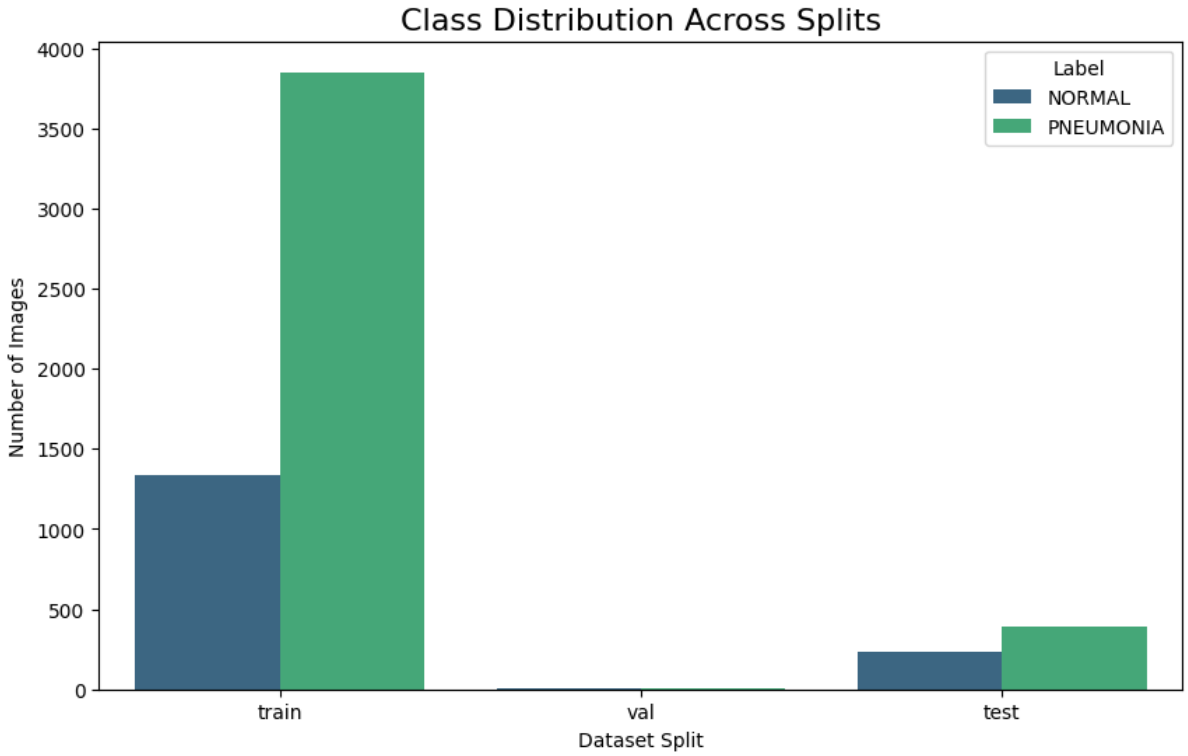


Рисунок 3.1 – Розподіл кількості зображень за вибірками та класами

На основі виявленого дисбалансу засобами функції compute_class_weight бібліотеки scikit-learn зі стратегією 'balanced' було розраховано вагові коефіцієнти класів: для класу NORMAL (індекс 0) отримано коефіцієнт 1,9366, для класу PNEUMONIA (індекс 1) – коефіцієнт 0,6740. Співвідношення цих коефіцієнтів становить приблизно 2,9, що відповідає оберненому співвідношенню кількості зображень класів у навчальній вибірці (3850 / 1340 ≈ 2,87) і означає, що під час навчання моделі помилкова класифікація одного зображення класу NORMAL штрафується приблизно в 2,9 раза сильніше, ніж аналогічна помилка для класу PNEUMONIA, що компенсує чисельну переважність останнього.

Перевірка стратегії аугментації зображень була виконана шляхом застосування налаштованого генератора ImageDataGenerator до одного зразкового зображення класу PNEUMONIA та візуалізації п'яти отриманих варіантів. Результат, наведений на рисунку 3.2, підтверджує, що обрані параметри аугментації (поворот до 10°, масштабування до 10%, зсуви до 10%,

зміна яскравості в межах 90–110%) створюють помітну, але анатомічно правдоподібну варіативність зображень, не спотворюючи діагностично значущих структур легеневих полів.

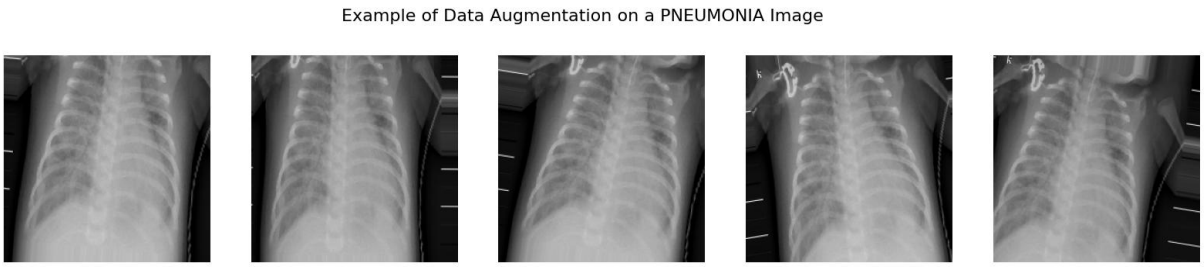


Рисунок 3.2 – Приклади аугментованих варіантів одного зображення класу PNEUMONIA

Додатково було здійснено вибіркову візуальну перевірку випадкових зображень кожного класу для підтвердження якості даних та коректності присвоєних міток. Приклади зображень класу NORMAL наведено на рисунку 3.3, а класу PNEUMONIA – на рисунку 3.4.

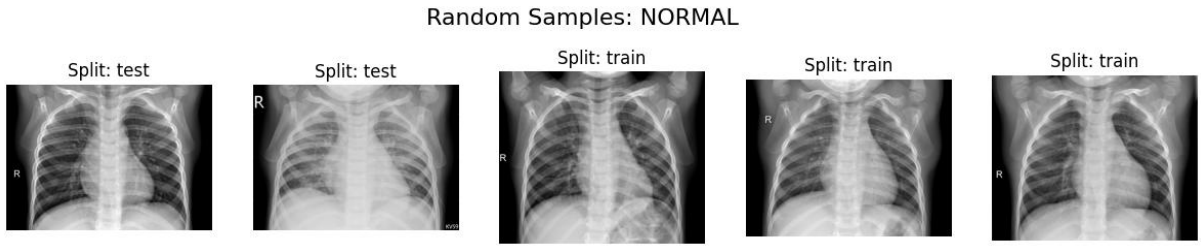


Рисунок 3.3 – Приклади випадкових зображень класу NORMAL

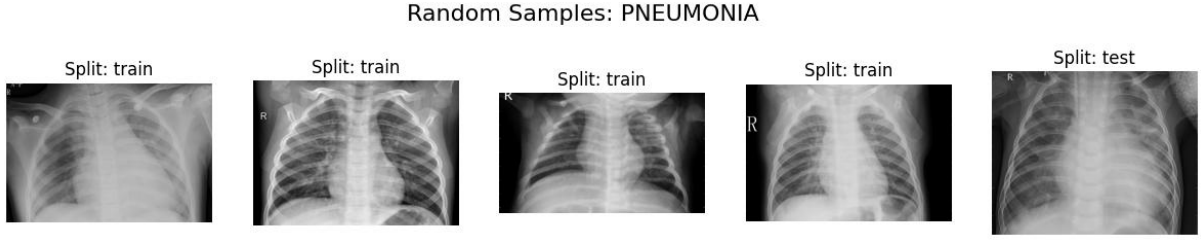


Рисунок 3.4 – Приклади випадкових зображень класу PNEUMONIA

Візуальний огляд підтвердив очікувані відмінності: на знімках класу PNEUMONIA спостерігаються характерні зони затемнення (інфільтрати) у легеневих полях, тоді як на знімках класу NORMAL легеневі поля мають рівномірну прозорість без виражених патологічних утворень.

3.4 Результати виконання модуля навчання моделей

На основі очищеного набору метаданих було сформовано три генератори даних: навчальний (train), що охопив 5190 зображень з аугментацією та перемішуванням; валідаційний (val) обсягом 16 зображень без аугментації; та тестовий (test) обсягом 618 зображень, також без аугментації. Сума зображень у трьох генераторах (5190 + 16 + 618 = 5824) повністю відповідає обсягу очищеного набору даних, що підтверджує коректність розподілу зображень за вибірками на етапі формування генераторів.

Кожна з трьох архітектур (MobileNetV2, VGG16, DenseNet121) була навчена за уніфікованою процедурою трансферного навчання, описаною в підрозділі 2.6, з максимальною кількістю епох 20 та механізмом дострокової зупинки за відсутності покращення метрики val_auc протягом 5 послідовних епох. Динаміку зміни валідаційної метрики AUC протягом навчання для всіх трьох моделей наведено на рисунку 3.5.

Графік на рисунку 3.5 демонструє суттєві відмінності у траєкторіях навчання моделей. Криві MobileNetV2 та DenseNet121 досить швидко досягають високих значень валідаційної AUC (близьких до 1,0) і надалі коливаються в цьому діапазоні, тоді як крива VGG16 демонструє нестабільну, значно менш переконливу динаміку. Слід зазначити, що через украй малий обсяг валідаційної вибірки (лише 16 зображень) поодинокі коливання метрики val_auc між епохами є природним явищем.

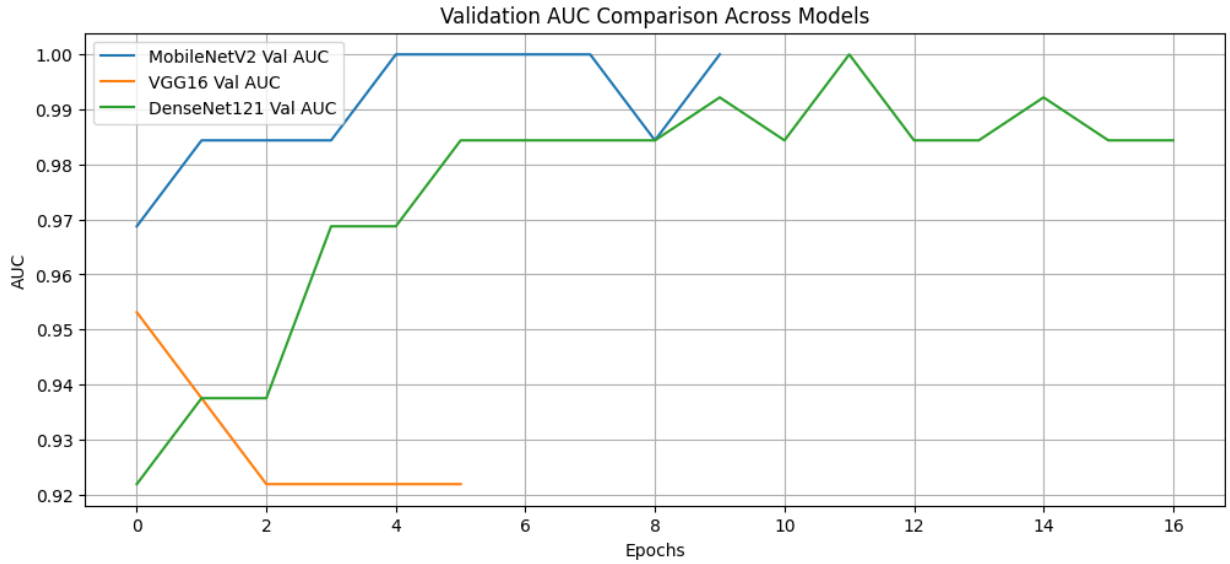


Рисунок 3.5 – Динаміка валідаційної метрики AUC протягом навчання трьох моделей

На такій малій вибірці навіть зміна класифікації одного-двох зображень може призводити до різких стрибків значення AUC, включно з досягненням теоретичного максимуму 1,0; тому остаточні висновки щодо якості моделей робилися виключно на основі результатів оцінювання на значно більшій та незалежній тестовій вибірці (618 зображень), розглянутих у підрозділі 3.5.

3.5 Результати оцінювання моделей на тестовій вибірці

Після завершення навчання для кожної з трьох моделей було завантажено найкращі збережені ваги (за критерієм максимальної валідаційної AUC) та виконано оцінювання на відкладеній тестовій вибірці обсягом 618 зображень, яка не використовувалася на жодному з попередніх етапів. Підсумкові значення чотирьох ключових метрик якості для кожної моделі наведено в таблиці 3.2.

Таблиця 3.2 – Підсумкові метрики якості моделей на тестовій вибірці

Модель	**AUC**	**Accuracy**	**Precision (PNEUMONIA)**	**Recall (PNEUMONIA)**
MobileNetV2	0,9674	0,9094	0,9344	0,9199
VGG16	0,9122	0,4434	1,0000	0,1111
DenseNet121	0,9538	0,8803	0,9384	0,8656

Для деталізованого аналізу характеру помилок кожної моделі додатково побудовано матриці помилок (confusion matrix), що відображають кількість істинно позитивних (TP), істинно негативних (TN), хибно позитивних (FP) та хибно негативних (FN) прогнозів для класу PNEUMONIA. Матриці помилок для всіх трьох моделей наведено на рисунку 3.6.

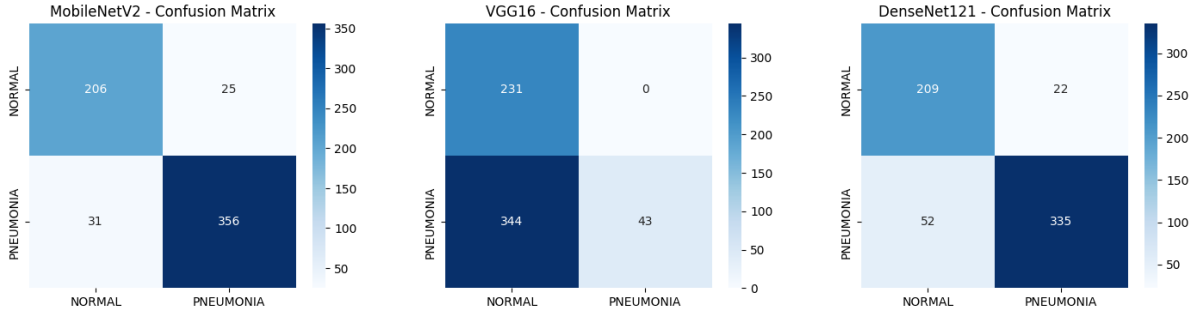


Рисунок 3.6 – Матриці помилок моделей MobileNetV2, VGG16 та DenseNet121 (зліва праворуч)

Аналіз матриці помилок MobileNetV2 (TN=206, FP=25, FN=31, TP=356) свідчить про збалансований характер помилок моделі: кількість хибно позитивних і хибно негативних прогнозів є відносно невеликою й приблизно зіставною, що підтверджує високі значення як точності (precision), так і повноти (recall) для цієї моделі. Натомість матриця помилок VGG16 (TN=231, FP=0, FN=344, TP=43) демонструє патологічну поведінку: модель практично завжди прогнозує клас NORMAL незалежно від фактичного вхідного зображення, про що свідчить нульова кількість хибно позитивних прогнозів у поєднанні з надзвичайно високою кількістю хибно негативних (344 з 387 фактичних випадків PNEUMONIA класифіковано неправильно). Це пояснює

формально ідеальне значення precision (1,0000) – модель просто не допускає жодного позитивного прогнозу, окрім дійсно правильних, – за вкрай низького значення recall (0,1111), що робить таку модель практично непридатною для медичної діагностики, оскільки переважна більшість фактичних випадків захворювання залишається невиявленою. Матриця помилок DenseNet121 (TN=209, FP=22, FN=52, TP=335) посідає проміжне положення: модель демонструє хорошу здатність виявляти випадки PNEUMONIA, проте дещо гіршу за MobileNetV2 повноту через більшу кількість хибно негативних прогнозів.

3.6 Порівняльний аналіз результатів та обґрунтування вибору фінальної моделі

Зіставлення результатів, наведених у таблиці 3.2, показує, що модель MobileNetV2 досягає найвищих значень за трьома з чотирьох розглянутих метрик (AUC, accuracy, recall) і близьке до найвищого значення за метрикою precision, поступаючись лише формально ідеальному, але фактично непридатному показнику VGG16. Модель DenseNet121 демонструє стабільно високі результати за всіма метриками, поступаючись MobileNetV2 на кілька відсоткових пунктів, і може розглядатися як надійна альтернатива. Модель VGG16 виявилася найменш придатною для розв'язання поставленої задачі: ймовірною причиною такого результату є поєднання порівняно високого значення швидкості навчання (learning_rate=0,001), застосованого уніфіковано до всіх трьох архітектур, з відсутністю в архітектурі VGG16 механізмів стабілізації градієнтного потоку, притаманних сучаснішим архітектурам MobileNetV2 (інвертовані залишкові з'єднання) та DenseNet121 (щільні з'єднання з повторним використанням ознак, докладніше розглянуті в підрозділах 1.4 та 1.6), через що модель VGG16, ймовірно, застрягла в незадовільному локальному рішенні, що зводиться до домінантної стратегії прогнозування найчисельнішого класу навчальної вибірки.

З огляду на критичну важливість метрики recall у задачах медичної діагностики (адже хибно негативний прогноз означає невиявлений випадок захворювання, що є значно небезпечнішим наслідком, ніж хибно позитивний прогноз, який лише призводить до додаткового, але не критичного обстеження), а також з урахуванням найвищого загального значення AUC і обчислювальної ефективності архітектури, розглянутої в підрозділі 1.4 (значно менша кількість параметрів і обчислювальних операцій порівняно з VGG16 та DenseNet121 завдяки використанню глибинно-роздільних згорток), фінальною моделлю для розв'язання задачі класифікації рентгенограм органів грудної клітки обрано MobileNetV2. Саме ця модель забезпечує оптимальне поєднання високої діагностичної якості та практичної придатності для розгортання в умовах обмежених обчислювальних ресурсів, що є важливою перевагою для потенційного практичного застосування в медичних закладах.

3.7 Висновки до третього розділу

У третьому розділі представлено результати практичного виконання розробленого програмного рішення на реальному наборі даних рентгенограм органів грудної клітки. Показано, що процедура попередньої обробки дозволила виявити та виключити 32 пошкоджені або повторювані зображення з вихідного набору обсягом 5856 файлів, сформувавши очищений набір метаданих обсягом 5824 зображення. Розвідувальний аналіз підтвердив наявність суттєвого дисбалансу класів (співвідношення приблизно 2,9:1 на користь класу PNEUMONIA) та обґрунтував застосування зважування класів і помірної аугментації даних.

Порівняльне навчання трьох архітектур трансферного навчання на тестовій вибірці обсягом 618 зображень виявило найвищу якість класифікації для моделі MobileNetV2 (AUC=0,9674, accuracy=0,9094, recall=0,9199), близьку до неї якість для DenseNet121 (AUC=0,9538) та незадовільний результат для VGG16, що практично завжди прогнозувала найчисельніший

клас (recall=0,1111). На основі комплексного аналізу метрик якості, з пріоритетом на максимізацію повноти виявлення випадків захворювання, обґрунтовано вибір моделі MobileNetV2 як фінального рішення поставленої задачі класифікації.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Аналіз небезпеки і шкідливості при розробці програмного забезпечення

Організація робочого місця розробника ПЗ впливає на його працездатність.

У своїй діяльності розробник використовує комп’ютер, пристрої збереження інформації, а тому є необхідність забезпечення зручного доступу до всіх технічних засобів. Тому в даному розділі докладніше розглянемо відомості про систему ергономічних норм і принципів організації робочого місця, на котрому проводяться роботи зі створення модуля збору статистики.

Під робочим місцем розуміється зона, оснащена необхідними технічними засобами, у якій відбувається трудова діяльність виконавця або групи виконавців, які спільно виконують одну роботу або операцію.

Організація робочого місця полягає у виконанні заходів, які забезпечують безпечний і раціональний трудовий процес і ефективне використання знарядь та предметів праці, що підвищує продуктивність праці і знижує стомлюваність працівника.

Організація робочого місця залежить від характеру розв'язуваних задач і особливостей предметно-просторового оточення, що визначають робоче положення тіла і можливість пауз для відпочинку, типи і способи засобів відображення і керування, необхідність у засобах захисту, спецодягу, простору для налагодження і ремонту устаткування.

Одним з компонентів діяльності на робочому місці є робочі рухи. Їхня раціональна організація створює умови для зниження стомлення, резерви для підвищеної працездатності. Просторові характеристики руху оператора визначаються траєкторіями руху і розмірами моторного поля (зони досяжності).

При організації робочого місця необхідно забезпечити нормальні умови огляду. Зону огляду описує кут, вершина якого знаходиться в центрі ока, а сторони складають границі, в яких людина при фіксованому положенні голови й ока добре розрізняє їхнє місцезнаходження.

У горизонтальній площині цей кут складає 300 – 400. При організації робочого місця кут огляду можна взяти 500 – 600, включаючи зону менш ясного огляду. Допустимий кут огляду по горизонталі 900. У вертикальній площині оптимальний кут огляду 100 вверх і 300 вниз від лінії погляду, а допустимий 300 вгору і 400 вниз від лінії погляду.

Щоб зберегти нормальну гостроту зору, робочу поверхню розташовують від очей на відстані від 0,3 м до 0,75 м. Робочі меблі повинні бути зручними для виконання робочих операцій. В даному випадку робочий стіл є основним устаткуванням. Особливо важливе значення має висота столу, його конструкція, яка повинна передбачати шухляди для розміщення інструментів, документації.

Важливе значення має конструкція робочих крісел. Погано підібрані крісла можуть бути причиною надмірної стомлюваності.

Нахил і висота крісла повинні регулюватися відповідно до висоти робочої поверхні і росту працюючого. Рекомендована ширина крісла 370 – 400 мм, глибина 370 – 420 мм, висота спинки 370 – 1000 мм від рівня крісла. Для розміщення ніг необхідно передбачити вільний простір під робочою площиною.

Праця людини, що протікає в умовах надмірного нервово-емоційного напруження, довготривалих статичних навантажень, обмеженої рухової активності призводить до неврозів, відхилень у психіці, захворювань опорно-рухового апарату, серцево-судинної системи тощо. Комп'ютери, телебачення, системи зв'язку та інші засоби, що використовують досягнення радіоелектроніки, є генераторами цілої низки електромагнітних випромінювань, вплив яких на організм людини ще не зовсім вивчений.

З широким впровадженням автоматизації та комп'ютеризації виникла потреба врахування психологічних можливостей людини, таких як швидкість реакції, особливості пам'яті та уваги, емоційний стан та ін. Поява операторської діяльності призвела до суттєвих змін у фаховій структурі праці. Зменшились фізична важкість праці, ризик виробничого травматизму, однак разом з тим, на працюючу людину посилюється вплив нових, раніше не відомих чи мало вивчених несприятливих виробничих факторів фізичного, хімічного і особливо психофізіологічного характеру.

Проте, розвиток сучасної обчислювальної техніки відбувається не лише у бік поктащення її технічних параметрів, але також звертається увага безпеку використання цієї техніки людиною шляхом зменшення потужності випромінювачів, зменшенням рівня випромінювання з моніторів, зменшення напруг живлення, покращення ергономічних характеристик.

Таким чином, в розділі з охорони праці виконано огляд питань безпечної роботи при створенні сайту та встановлено, що умови такої роботи відповідають вимогам з охорони праці, які застосовуються в галузі інформаційних технологій.

4.2 Інформаційно-психологічні небезпеки

Сучасні реалії постіндустріального суспільства, зумовлені значним ростом інформації, відкривають ще одну сферу життєдіяльності людини – інформаційну. Сучасні засоби комунікації і обробки інформації створили принципово нові умови існування людини, що зумовило появу грандіозного проекту об'єднання національних інформаційних і телекомунікаційних структур в глобальну інформаційну інфраструктуру.

Життєдіяльність людини реалізується одночасно зі світом природи і у специфічному для людського суспільства інформаційному середовищі, що має свої закономірності розвитку і функціонування. Інформаційна сфера стає такою ж важливою складовою суспільного життя, як економічна, виробнича,

побутова, політична, військова та ін. Нові інформаційні технології, засоби масової комунікації багатократно підсилили можливості впливу на свідомість і підсвідомість як окремої людини, так і на великі групи людей та населення країни загалом.

Інформаційна сфера – сукупність таких елементів:

– об'єкти інформаційної взаємодії чи впливу;

– особисто інформація, призначена для використання суб'єктами інформаційної сфери;

– інформаційна інфраструктура, що забезпечує можливість здійснення обміну інформацією між суб'єктами;

– суспільні відносини, що складаються у зв'язку з формуванням, переданням, розповсюдженням і збереженням інформації.

Особистість, активний соціальний суб'єкт, його психіка піддаються безпосередньому впливу інформаційних чинників (передумов, що чинять опір чи утруднюють формування і функціонування адекватної інформаційно-орієнтуючої основи суспільної поведінки людини (життєдіяльності у суспільстві)), які трансформуються, через його поведінку, діяльність (бездіяльність), здійснюють деструктивний, дисфункційний вплив на його життєдіяльність.

До основних загроз інформаційно-психологічної безпеки відносять можливість настання негативних наслідків для суб'єктів, що піддаються інформаційно-психологічному впливу, які виражаються в таких формах:

– нанесення шкоди здоров'ю людини;

– блокування на неусвідомленому рівні волі, волевиявлення людини, штучне прививття їй синдрому залежності;

– втрата здатності до політичної, культурної, моральної самоіндентифікації людини;

– маніпуляція суспільною свідомістю;

– руйнування єдиного інформаційного і духовного простору України, традиційних устроїв суспільства і суспільної моральності, а також порушення інших життєво важливих інтересів особистості, суспільства, держави.

Наприклад, культ жорстокості, насильства, порнографії, розбещеності тощо, які пропагують у засобах масової інформації, друкованих виданнях, комп'ютерних іграх, мережі Інтернет веде до неусвідомленого бажання у підлітків і молоді, а також дорослих з нестійкою психікою, копіювати запропоновані моделі поведінки. Цей вид пропаганди знижує рівень порогових обмежень і правових заборон, що поряд з іншими умовами відкриває шлях для багатьох правопорушень. Це своєю чергою наносить непоправну шкоду не тільки окремій особистості, але й суттєві збитки національним інтересам країни.

Отже, джерелом інформаційно-психологічної небезпеки є та частина інформаційного середовища, яка через визначені причини неадекватно відображає реалії, вводить в оману людину, засліплює її ілюзією.

Інформаційно-психологічні загрози зумовлені розробкою, виготовленням, розповсюдженням та використанням суб'єктами негативних інформаційно-психологічних впливів, спеціальних засобів і методів такого впливу.

Концепція інформаційно-психологічної безпеки.

Сучасне розуміння безпеки в контексті врахування відношення інтересів особистості, суспільства і держави висуває завдання розгляду нового аспекту цієї проблеми – безпеки в інформаційній сфері життєдіяльності людини, тобто інформаційно-психологічної безпеки.

В інформаційному середовищі, що є складовим системним утворенням, виділяється процесуальна складова як найбільш динамічна і змінна її частина – інформаційно-комунікативні процеси, які активно впливають на індивідуальну, групову і суспільну психологію (індивідуальну, групову, масову свідомість). Маніпулюючи станом інформаційного середовища,

змінюється стан духовної сфери суспільства, деформація і деструктивні зміни якої у формі психоемоційної і соціальної напруженості, спотворених норм і неадекватних соціальних стереотипів і установок, оманливих і неприродних орієнтацій та цінностей. Це своєю чергою впливає на стан і процеси у всіх основних сферах суспільного життя, в тому числі політичній і економічній.

Вперше у пострадянському просторі про проблему інформаційно-психологічної безпеки було зазначено в листопаді 1995 р. на науково-практичній конференції, організованій Інститутом психології Російської академії наук. На цій та подальших конференціях було розкрито роль знання технологій інформаційно-психологічного впливу, метою якого є маніпуляція, для вироблення напрямів реформування психологічного захисту особистості і особистої інформаційно-психологічної безпеки.

Інформаційно-психологічну безпеку особистості визначають такими основними причинами.

Зростання тиску інформаційного середовища визначає необхідність формування нових механізмів та засобів виживання людини як особистості й активного соціального суб'єкта у сучасному суспільстві.

Взаємодія психіки людини з інформаційним середовищем відрізняється якісною специфікою і не має аналогів у комунікації інших біологічних, технічних, соціальних і соціотехнічних структур.

Основною і центральною "мішенню" інформаційного впливу є людина, її психіка.

Отже, інформаційно-психологічну безпеку можливо розглядати як стан захищеності особистості, різних соціальних груп і об'єднань людей від дій, впливів, які здатні проти їхньої волі і бажання змінити психічні стани та психологічні характеристики людини, модифікувати її поведінку і обмежувати свободу вибору, зумовило потребу переосмислення інформаційної взаємодії, а також деяких інших соціально-психологічних процесів і явищ у сучасному суспільстві.

Інформаційно-психологічна безпека – стан захищеності окремих осіб чи груп осіб від негативних інформаційно-психологічних впливів і пов'язаних з цим інших життєво важливих інтересів особистості, суспільства, держави в інформаційному середовищі.

Негативний інформаційно-психологічний вплив – процес зміни психічних станів і характеристик людей під впливом інформаційно-комунікативних процесів як динамічного компонента інформаційного середовища. Цей вплив спрямований на людину чи групу осіб (у тому числі без їхньої згоди) з метою примусу до визначеної поведінки, оцінки ситуації, керування та корекції індивідуальної та колективної свідомості. Він здійснюється з використанням спеціальних засобів і методів впливу на психіку людини, унаслідок чого він приводить до негативних наслідків для особистості, суспільства і держави.

Спеціальні засоби впливу – технічні і програмні засоби, що використовують для використання з метою негативного інформаційно-психологічного впливу на людину чи групу людей.

Спеціальні методи впливу – послідовність прийомів впливу на психіку людини, використання яких приводить до негативних наслідків для особистості, суспільства та держави.

Головним об'єктом забезпечення інформаційно-психологічної безпеки в інформаційному середовищі у сфері індивідуальної безпеки є усвідомлення інформації, здатність людини адекватно сприймати навколишню дійсність, своє місце в зовнішньому світі, формувати відповідно до свого життєвого досвіду визначені переконання і приймати стосовно них рішення.

Інформаційно-психологічна безпека має спиратися на стандарти інформаційно-психологічної безпеки – затверджені у визначеному порядку інформаційно-психологічного впливу, який не викликає негативних наслідків для психіки людини.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено та практично реалізовано повний програмний конвеєр для автоматизованої класифікації рентгенограм органів грудної клітки за двома класами – NORMAL (норма) та PNEUMONIA (пневмонія) – із застосуванням методів глибокого навчання та трансферного навчання на основі сучасних згорткових нейронних мереж. Поставлені на початку роботи завдання виконано в повному обсязі, що дозволяє сформулювати такі основні висновки.

В першу чергу було проаналізовано теоретичні основи глибокого навчання, згорткових нейронних мереж та концепції трансферного навчання, а також детально розглянуто архітектурні особливості трьох обраних для порівняння моделей – MobileNetV2, VGG16 та DenseNet121. Показано, що ефективність архітектури MobileNetV2 ґрунтується на використанні глибинно-роздільних згорток та інвертованих залишкових блоків, тоді як висока представницька здатність DenseNet121 пояснюється механізмом щільних з'єднань і повторного використання ознак, а класична архітектура VGG16, незважаючи на простоту й історичну значущість, потребує значно більшої кількості параметрів для досягнення зіставної якості.

Наступним кроком роботи була реалізація та обґрунтування трирівневого програмного конвеєра обробки даних, що охоплює:

– модуль попередньої обробки із застосуванням MD5-хешування та перевірки цілісності файлів засобами Pillow;

– модуль розвідувального аналізу даних із розрахунком вагових коефіцієнтів класів та формуванням стратегії аугментації, адаптованої до специфіки медичних рентгенівських зображень (зокрема, обґрунтовано виключення горизонтального відображення);

– модуль побудови, навчання й оцінювання моделей із застосуванням механізмів автоматичної дострокової зупинки навчання (EarlyStopping) і збереження найкращих контрольних точок (ModelCheckpoint).

Далі на основі очищеного набору даних обсягом 5824 зображення виконано порівняльне навчання трьох моделей трансферного навчання та проведено їх оцінювання за чотирма ключовими метриками якості – точністю (accuracy), площею під ROC-кривою (AUC), точністю класифікації позитивного класу (precision) та повнотою (recall). Емпірично встановлено, що модель MobileNetV2 демонструє найвищу загальну якість класифікації (AUC=0,9674, accuracy=0,9094, recall=0,9199), модель DenseNet121 посідає близьке друге місце (AUC=0,9538), тоді як модель VGG16 за обраних умов навчання виявилася практично непридатною для розв'язання задачі (recall=0,1111), що ілюструє важливість урахування архітектурних особливостей моделі під час підбору гіперпараметрів навчання, зокрема швидкості навчання.

В кінці роботи з огляду на критичну важливість мінімізації хибно негативних діагностичних рішень у медичній практиці, обґрунтовано вибір моделі MobileNetV2 як фінального рішення поставленої задачі – не лише через найвищі показники якості за більшістю метрик, а й через суттєву обчислювальну ефективність цієї архітектури, що є важливим практичним фактором для потенційного розгортання системи в умовах обмежених обчислювальних ресурсів медичних закладів.

Отримані в роботі результати підтверджують практичну доцільність застосування методів трансферного навчання для розв'язання задач медичної діагностики на основі зображень навіть за відносно обмеженого обсягу доступних навчальних даних, а розроблений програмний конвеєр може бути використаний як основа для подальшого розширення функціональності, зокрема для багатокласової класифікації різних типів патологій легень, інтеграції механізмів інтерпретованості моделі (наприклад, методу Grad-CAM для візуалізації областей зображення, що найбільше впливають на рішення моделі), а також для проспективної клінічної валідації на додаткових, незалежних наборах рентгенівських знімків з інших медичних закладів.

СПИСОК ЛІТЕРАТУРИ

1. Гудз Я. М. Застосунок для перенесення обличчя на відео на основі нейронних мереж : робота на здобуття кваліфікаційного ступеня бакалавра : спец. 122 — комп'ютерні науки / наук. кер. Л. П. Матійчук. Тернопіль : ТНТУ, 2025. 58 с. URL: https://elartu.tntu.edu.ua/handle/lib/49405.

2. Мельник О. О. Розробка архітектури базової нейронної мережі для сегментації меж та границь на зображеннях : робота на здобуття кваліфікаційного ступеня бакалавра / наук. кер. О. М. Денисюк. Тернопіль : ТНТУ, 2025. 65 с. URL: https://elartu.tntu.edu.ua/handle/lib/49054.

3. Bokhonko A., Melnykova N., Patereha Y. Comparative analysis of data augmentation methods for image modality. Scientific journal of the Ternopil National Technical University. 2024. Vol. 113, No. 1. P. 16–26. DOI: https://doi.org/10.33108/visnyk_tntu2024.01.016.

4. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2018. – P. 4510–4520.

5. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // International Conference on Learning Representations (ICLR). – 2015. – 14 p.

6. Huang G., Liu Z., van der Maaten L., Weinberger K. Q. Densely Connected Convolutional Networks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 4700–4708.

7. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge, MA : MIT Press, 2016. – 800 p.

8. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems (NeurIPS). – 2012. – Vol. 25. – P. 1097–1105.

9. Kermany D. S., Goldbaum M., Cai W. et al. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning // Cell. – 2018. – Vol. 172, № 5. – P. 1122–1131.

10. Pan S. J., Yang Q. A Survey on Transfer Learning // IEEE Transactions on Knowledge and Data Engineering. – 2010. – Vol. 22, № 10. – P. 1345–1359.

11. Pedregosa F., Varoquaux G., Gramfort A. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. – 2011. – Vol. 12. – P. 2825–2830.

12. Chollet F. Deep Learning with Python. – 2nd ed. – Shelter Island, NY : Manning Publications, 2021. – 504 p.

13. Mooney P. Chest X-Ray Images (Pneumonia) [Електронний ресурс] : Kaggle Dataset. – Режим доступу: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia (дата звернення: 28.01.2026).

14. Стручок, В. С., Стручок, О. С., & Мудра, Д. В. (2017). Навчальний посібник до написання розділу дипломного проекту та дипломної роботи ″Безпека в надзвичайних ситуаціях ″для студентів всіх спец. денної, заочної (дистанційної) та екстернатної форм навчання.

15. Стручок, В. С. (2022). Техноекологія та цивільна безпека. Частина "Цивільна безпека". Навчальний посібник.

16. Жидецький, В. Ц., Джигирей, В. С., & Мельников, О. В. (2000). Основи охорони праці. Львів: Афіша, 350, 132-136.

17. Навакатікян О. О., Кальниш В. В., & Стрюков С. М. (1997). Охорона праці користувачів комп'ютерних відеодисплейних терміналів. О. Навакатікян.

ДОДАТКИ

ДОДАТОК А

Програмний код

Chest X-Ray Pneumonia Classification – Preprocessing

The dataset comes from [Kaggle – Chest X-Ray Images (Pneumonia)].

The dataset is **already split** into `train`, `val`, and `test` sets.

- `train/` → Model training
- `val/` → Hyperparameter tuning
- `test/` → Final evaluation

```
#Import Libraries
import os
import hashlib
import pandas as pd
from tqdm import tqdm
from PIL import Image # Using Pillow to quickly verify images

print("☑ Libraries imported.")
```

```
✅ Libraries imported.
```

Step 1: Collect Image Paths & Labels

We'll start by looping through the data directory to gather the filepath and corresponding label for every image.

```
# Define base directory
base_dir = "Data/chest_xray"
splits = ["train", "val", "test"]
categories = ["NORMAL", "PNEUMONIA"]

# Build a list of dictionaries with image data
image_data = []
for split in splits:
    for category in categories:
        folder_path = os.path.join(base_dir, split, category)
        # Check if folder exists to prevent errors
        if not os.path.isdir(folder_path):
            continue
        for img_file in os.listdir(folder_path):
            image_data.append({
                "filepath": os.path.join(folder_path, img_file),
                "label": category,
                "split": split
            })
```

```
df = pd.DataFrame(image_data)
print(f"Found {len(df)} total image paths.")
df.head()
```

```
Found 5856 total image paths.
```

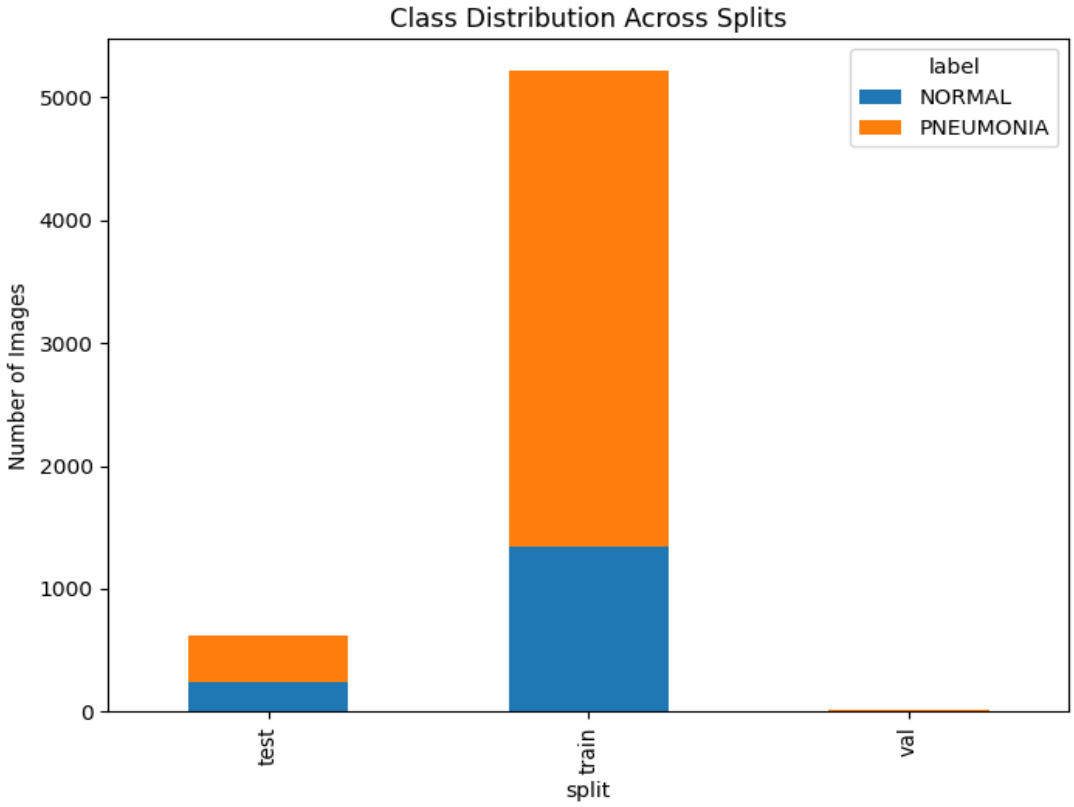
	filepath	label	split
0	Data/chest_xray\train\NORMAL\IM-0115-0001.jpeg	NORMAL	train
1	Data/chest_xray\train\NORMAL\IM-0117-0001.jpeg	NORMAL	train
2	Data/chest_xray\train\NORMAL\IM-0119-0001.jpeg	NORMAL	train
3	Data/chest_xray\train\NORMAL\IM-0122-0001.jpeg	NORMAL	train
4	Data/chest_xray\train\NORMAL\IM-0125-0001.jpeg	NORMAL	train

Step 2: Class Distribution

Before cleaning, let's check how many NORMAL vs. PNEUMONIA images exist in each split.

```
import matplotlib.pyplot as plt

# Count images in each split/category
counts = df.groupby(["split", "label"]).size().unstack()
print(counts)

# Visualization
counts.plot(kind="bar", stacked=True, figsize=(8,6))
plt.title("Class Distribution Across Splits")
plt.ylabel("Number of Images")
plt.show()
```

label	NORMAL	PNEUMONIA
split		
test	234	390
train	1341	3875
val	8	8



Step 3: Remove Duplicates

Now, we'll ensure every entry in our dataframe points to a valid, unique image.

- **Corrupted Files**: We'll try to open each image. If it fails, we mark it for removal.
- **Duplicate Files**: We'll generate an **MD5 hash** (a unique signature) for each image file to find and remove exact duplicates.

```
def get_file_hash(filepath):
    """Calculates the MD5 hash of a file."""
    with open(filepath, "rb") as f:
        return hashlib.md5(f.read()).hexdigest()

valid_images = []
hashes = set()
for index, row in tqdm(df.iterrows(), total=df.shape[0], desc="Verifying Imag
es"):
    filepath = row['filepath']
    try:
        # 1. Check if it's a valid image file that can be opened
        img = Image.open(filepath)
        img.verify() # Verify that it is, in fact, an image

        # 2. Check for duplicates using hashing
        file_hash = get_file_hash(filepath)
        if file_hash in hashes:
            continue # Skip if it's a duplicate

        hashes.add(file_hash)
        valid_images.append(row)
```

```
    except Exception as e:
        print(f"Skipping corrupted file: {filepath} due to {e}")

# Create a new, clean DataFrame
clean_df = pd.DataFrame(valid_images)

print(f"\nOriginal image count: {len(df)}")
print(f"Clean image count: {len(clean_df)}")
print(f"Removed {len(df) - len(clean_df)} corrupted or duplicate images.")
```

```
Verifying Images: 100%|██████████| 5856/5856 [01:39<00:00, 58.58it/s]


Original image count: 5856
Clean image count: 5824
Removed 32 corrupted or duplicate images.
```

Step 4: Save Clean Metadata

This is the **final and most important step**. We save our clean DataFrame to a CSV file. This single file will be the input for our EDA and Model Building notebooks, ensuring a reliable and reproducible workflow.

```
# Save the clean metadata to a CSV file
output_path = "Data/clean_metadata.csv"
clean_df.to_csv(output_path, index=False)

print(f"☑ Clean metadata saved to {output_path}")
clean_df.head()
```

```
✅ Clean metadata saved to Data/clean_metadata.csv
```

	filepath	label	split
0	Data/chest_xray\train\NORMAL\IM-0115-0001.jpeg	NORMAL	train
1	Data/chest_xray\train\NORMAL\IM-0117-0001.jpeg	NORMAL	train
2	Data/chest_xray\train\NORMAL\IM-0119-0001.jpeg	NORMAL	train
3	Data/chest_xray\train\NORMAL\IM-0122-0001.jpeg	NORMAL	train
4	Data/chest_xray\train\NORMAL\IM-0125-0001.jpeg	NORMAL	train

Chest X-Ray Pneumonia Classification – Exploration

We are using the **cleaned metadata file** generated in the preprocessing step:
`Data/clean_metadata.csv`

This ensures we work only with **reliable and standardized images**.

Import Libraries & Load Metadata

```
#Import Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array, Imag
eDataGenerator
from sklearn.utils import class_weight

# Load the metadata we created in the preprocessing step
df = pd.read_csv("Data/clean_metadata.csv")

print(f"☑ Metadata loaded successfully. Total images: {len(df)}")
df.head()
```

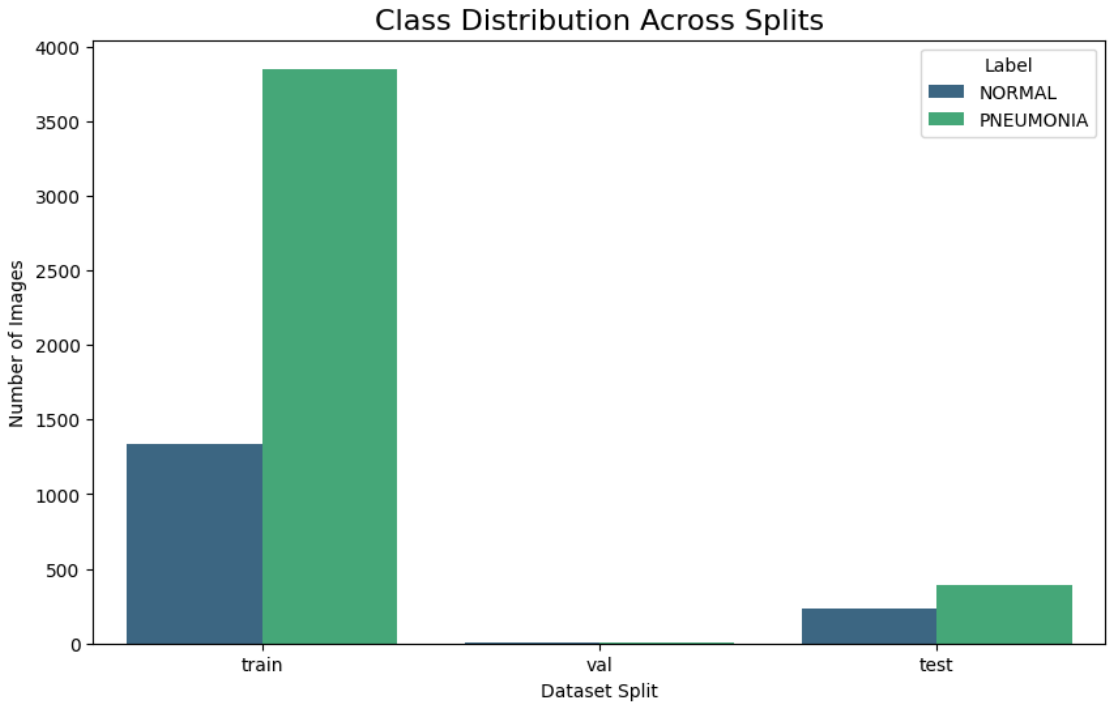
```
✅ Metadata loaded successfully. Total images: 5824
```

	filepath	label	split
0	Data/chest_xray\train\NORMAL\IM-0115-0001.jpeg	NORMAL	train
1	Data/chest_xray\train\NORMAL\IM-0117-0001.jpeg	NORMAL	train
2	Data/chest_xray\train\NORMAL\IM-0119-0001.jpeg	NORMAL	train
3	Data/chest_xray\train\NORMAL\IM-0122-0001.jpeg	NORMAL	train
4	Data/chest_xray\train\NORMAL\IM-0125-0001.jpeg	NORMAL	train

Analyze Class Distribution

A balanced dataset is key for an unbiased model. Let's check the number of `NORMAL` vs. `PNEUMONIA` cases across our `train`, `val`, and `test` splits.

```
# Count samples across splits and labels
# Plot the distribution
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='split', hue='label', palette='viridis')
plt.title("Class Distribution Across Splits", fontsize=16)
plt.ylabel("Number of Images")
plt.xlabel("Dataset Split")
plt.legend(title='Label')
plt.show()

# Print the exact counts
print(df.groupby(['split', 'label']).size().unstack(fill_value=0))
```



label	NORMAL	PNEUMONIA
split		
test	231	387
train	1340	3850
val	8	8

Observation:

The training set has a severe **class imbalance**. There are far more PNEUMONIA images than NORMAL ones. We must address this.

Plan for Imbalance: Calculate Class Weights

To prevent our model from being biased towards the majority class, we'll use class weights. This technique penalizes the model more for misclassifying the minority class (NORMAL).

```
# Isolate the training data to calculate weights
train_df = df[df['split'] == 'train']

# Calculate class weights using scikit-learn
weights = class_weight.compute_class_weight(
    'balanced',
    classes=np.unique(train_df['label']),
    y=train_df['label']
)

# Keras expects a dictionary mapping class indices to weights
# Let's map NORMAL to 0 and PNEUMONIA to 1
class_map = {'NORMAL': 0, 'PNEUMONIA': 1}
class_weight_dict = {class_map[label]: weight for label, weight in zip(np.uni
que(train_df['label']), weights)}

print("Labels mapped to indices:", class_map)
```

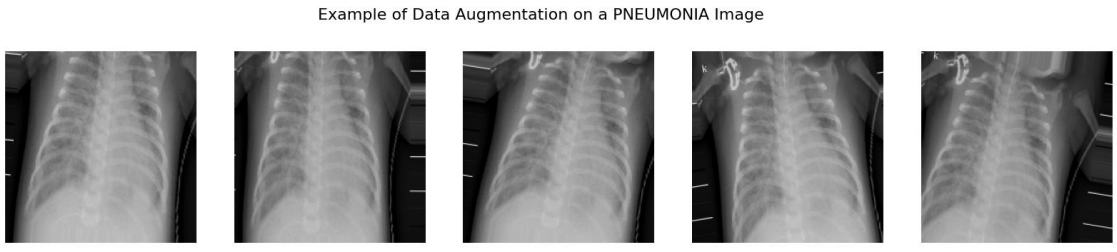
```
print("Calculated Class Weights:", class_weight_dict)
print("\nDuring training, mistakes on 'NORMAL' images will be penalized ~3.8x
more than mistakes on 'PNEUMONIA' images.")
```

```
Labels mapped to indices: {'NORMAL': 0, 'PNEUMONIA': 1}
Calculated Class Weights: {0: np.float64(1.9365671641791045), 1: np.float64(0
.674025974025974)}

During training, mistakes on 'NORMAL' images will be penalized ~3.8x more tha
n mistakes on 'PNEUMONIA' images.
```

Visualize Data Augmentations

Data augmentation artificially expands our training set by creating modified copies of images. This helps the model generalize better. Let's visualize what our chosen augmentations will look like.

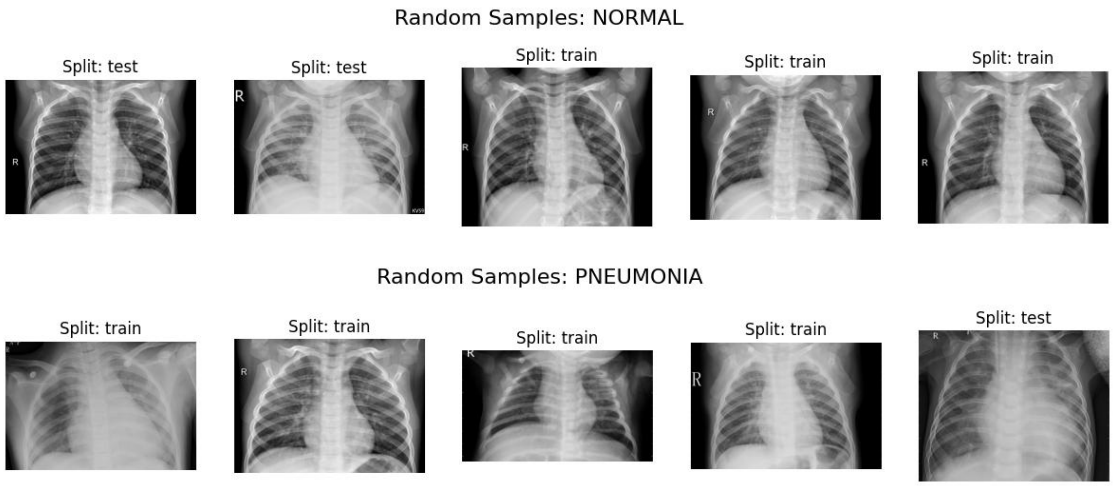
```
# Define our augmentation strategy in an ImageDataGenerator
aug_gen = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    brightness_range=[0.9, 1.1],
    horizontal_flip=False # Not anatomically correct for X-rays
)

# Pick one sample image to visualize
sample_path = df[df['label'] == 'PNEUMONIA']['filepath'].iloc[0]
img = load_img(sample_path, target_size=(224, 224))
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) # Add batch dimension

# Generate and plot 5 augmented versions of the sample image
plt.figure(figsize=(20, 4))
plt.suptitle("Example of Data Augmentation on a PNEUMONIA Image", fontsize=16
)
i = 0
for batch in aug_gen.flow(img_array, batch_size=1):
    plt.subplot(1, 5, i + 1)
    plt.imshow(batch[0] / 255.0)
    plt.axis('off')
    i += 1
    if i % 5 == 0:
        break
plt.show()
```



Final Sanity Check: Visualize Samples

Finally, let's look at a few random images from each class to ensure our file paths are correct and the labels make sense visually.

```
def show_samples(label, n=5):
    """Displays n random samples for a given label."""
    sample_df = df[df['label'] == label].sample(n)
    plt.figure(figsize=(15, 3))
    plt.suptitle(f'Random Samples: {label}', fontsize=16)
    for i, (_, row) in enumerate(sample_df.iterrows()):
        img = load_img(row["filepath"], color_mode="grayscale")
        plt.subplot(1, n, i + 1)
        plt.imshow(img, cmap="gray")
        plt.title(f"Split: {row['split']}")
        plt.axis("off")
    plt.show()

# Show samples for both classes
show_samples("NORMAL")
show_samples("PNEUMONIA")
```



Conclusion:

Our EDA confirms the dataset is clean and ready. We've identified a class imbalance, calculated weights to correct it, and planned our data augmentation strategy. We're now fully prepared for `03_Model_Building.ipynb`!

Chest X-Ray Pneumonia Classification – Model Comparison

Setup and Data Preparation

This step is the same as before. We set up our environment and create the data generators that will feed all three models. This ensures a fair comparison.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

# Import model architectures
from tensorflow.keras.applications import MobileNetV2, VGG16, DenseNet121
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification_report, confusion_matrix

# --- Constants ---
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
LEARNING_RATE = 0.001
EPOCHS = 20 # Max epochs, EarlyStopping will likely stop it sooner

# --- Load and Prepare DataFrames ---
df = pd.read_csv("Data/clean_metadata.csv")
df['label_idx'] = df['label'].map({'NORMAL': 0, 'PNEUMONIA': 1})

# --- Create Data Generators ---
train_datagen = ImageDataGenerator(
    rescale=1./255., rotation_range=10, zoom_range=0.1,
    width_shift_range=0.1, height_shift_range=0.1
)
val_test_datagen = ImageDataGenerator(rescale=1./255.)

# Train generator (using the 'train' split)
train_generator = train_datagen.flow_from_dataframe(
    dataframe=df[df['split'] == 'train'], x_col='filepath', y_col='label_idx'
,
    target_size=IMG_SIZE, class_mode='raw', batch_size=BATCH_SIZE, shuffle=Tr
ue, color_mode='rgb'
)

# Validation generator (using the 'val' split)
validation_generator = val_test_datagen.flow_from_dataframe(
    dataframe=df[df['split'] == 'val'], x_col='filepath', y_col='label_idx',
    target_size=IMG_SIZE, class_mode='raw', batch_size=BATCH_SIZE, shuffle=Fa
```

```
lse, color_mode='rgb'
)

# Test generator (using the 'test' split)
test_generator = val_test_datagen.flow_from_dataframe(
    dataframe=df[df['split'] == 'test'], x_col='filepath', y_col='label_idx',
    target_size=IMG_SIZE, class_mode='raw', batch_size=BATCH_SIZE, shuffle=Fa
lse, color_mode='rgb'
)

# Class weights calculated from our EDA
class_weight_dict = {0: 3.82, 1: 0.68}

print("☑ Setup complete. Data generators are ready.")
```

```
Found 5190 validated image filenames.
Found 16 validated image filenames.
Found 618 validated image filenames.
✅ Setup complete. Data generators are ready.
```

Model Training Function

To avoid repeating code, we'll create a single function that can build, compile, and train any of the models we give it.

```
def build_and_train_model(base_model_class, model_name):
    """
    Builds, compiles, and trains a transfer learning model.

    Args:
        base_model_class: The Keras base model class (e.g., MobileNetV2).
        model_name (str): A name for the model (e.g., "MobileNetV2").

    Returns:
        The trained model and its history.
    """
    print(f"--- Training {model_name} ---")

    # Build model
    base_model = base_model_class(weights='imagenet', include_top=False, inpu
t_shape=(224, 224, 3))
    base_model.trainable = False
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile model
    model.compile(
        optimizer=Adam(learning_rate=LEARNING_RATE),
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
    )

    # Callbacks
```

```
    checkpoint = ModelCheckpoint(f"best_{model_name}.keras", save_best_only=T
rue, monitor="val_auc", mode="max")
    early_stopping = EarlyStopping(monitor="val_auc", patience=5, mode="max",
restore_best_weights=True)

    # Train
    history = model.fit(
        train_generator,
        epochs=EPOCHS,
        validation_data=validation_generator,
        class_weight=class_weight_dict,
        callbacks=[checkpoint, early_stopping],
        verbose=1
    )

    print(f"--- Finished Training {model_name} ---\n")
    return model, history

# --- Train All Three Models ---
models_to_train = {
    "MobileNetV2": MobileNetV2,
    "VGG16": VGG16,
    "DenseNet121": DenseNet121
}

trained_models = {}
histories = {}

for name, model_class in models_to_train.items():
    model, history = build_and_train_model(model_class, name)
    trained_models[name] = model
    histories[name] = history
```

```
--- Training MobileNetV2 ---
Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
tions/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no
_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

c:\Users\Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\kera
s\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `Py
Dataset` class should call `super().__init__(**kwargs)` in its constructor. `
**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do
not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()

Epoch 1/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 148s 887ms/step - accuracy: 0.8376 - auc: 0.9620
- loss: 0.3769 - val_accuracy: 0.8750 - val_auc: 0.9688 - val_loss: 0.2765
Epoch 2/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 139s 853ms/step - accuracy: 0.9114 - auc: 0.9848
- loss: 0.2219 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2505
Epoch 3/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 137s 843ms/step - accuracy: 0.9214 - auc: 0.9874
- loss: 0.2014 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2150
Epoch 4/20
```

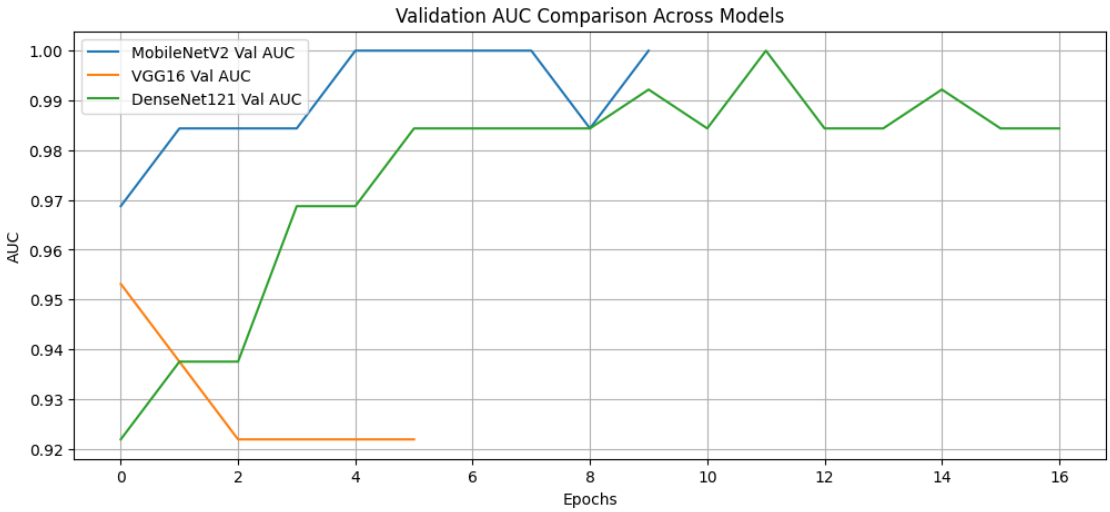
```
163/163 ━━━━━━━━━━━━━━━━━━━━ 138s 848ms/step - accuracy: 0.9331 - auc: 0.9895
- loss: 0.1781 - val_accuracy: 0.9375 - val_auc: 0.9844 - val_loss: 0.2010
Epoch 5/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 138s 847ms/step - accuracy: 0.9339 - auc: 0.9913
- loss: 0.1623 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.1938
Epoch 6/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 134s 818ms/step - accuracy: 0.9383 - auc: 0.9915
- loss: 0.1560 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.1992
Epoch 7/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 133s 817ms/step - accuracy: 0.9428 - auc: 0.9924
- loss: 0.1490 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2309
Epoch 8/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 132s 810ms/step - accuracy: 0.9447 - auc: 0.9919
- loss: 0.1477 - val_accuracy: 0.9375 - val_auc: 1.0000 - val_loss: 0.1828
Epoch 9/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 131s 805ms/step - accuracy: 0.9459 - auc: 0.9943
- loss: 0.1323 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2077
Epoch 10/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 132s 808ms/step - accuracy: 0.9495 - auc: 0.9936
- loss: 0.1294 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.1797
--- Finished Training MobileNetV2 ---

--- Training VGG16 ---
Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
tions/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Epoch 1/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 310s 2s/step - accuracy: 0.3168 - auc: 0.8499 -
loss: 0.8428 - val_accuracy: 0.6250 - val_auc: 0.9531 - val_loss: 0.5842
Epoch 2/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 308s 2s/step - accuracy: 0.6653 - auc: 0.9451 -
loss: 0.6512 - val_accuracy: 0.8125 - val_auc: 0.9375 - val_loss: 0.4895
Epoch 3/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 307s 2s/step - accuracy: 0.7715 - auc: 0.9489 -
loss: 0.5498 - val_accuracy: 0.8125 - val_auc: 0.9219 - val_loss: 0.4424
Epoch 4/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 307s 2s/step - accuracy: 0.8048 - auc: 0.9504 -
loss: 0.4907 - val_accuracy: 0.7500 - val_auc: 0.9219 - val_loss: 0.4057
Epoch 5/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 309s 2s/step - accuracy: 0.8241 - auc: 0.9521 -
loss: 0.4513 - val_accuracy: 0.7500 - val_auc: 0.9219 - val_loss: 0.3861
Epoch 6/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 321s 2s/step - accuracy: 0.8341 - auc: 0.9547 -
loss: 0.4210 - val_accuracy: 0.7500 - val_auc: 0.9219 - val_loss: 0.3787
--- Finished Training VGG16 ---

--- Training DenseNet121 ---
Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
tions/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5
29084464/29084464 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
Epoch 1/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 334s 2s/step - accuracy: 0.7102 - auc: 0.9083 -
loss: 0.5722 - val_accuracy: 0.8125 - val_auc: 0.9219 - val_loss: 0.3451
Epoch 2/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 293s 2s/step - accuracy: 0.8667 - auc: 0.9697 -
```

```
loss: 0.3270 - val_accuracy: 0.8125 - val_auc: 0.9375 - val_loss: 0.3063
Epoch 3/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 349s 2s/step - accuracy: 0.8917 - auc: 0.9770 -
loss: 0.2744 - val_accuracy: 0.8750 - val_auc: 0.9375 - val_loss: 0.2811
Epoch 4/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 356s 2s/step - accuracy: 0.9033 - auc: 0.9803 -
loss: 0.2527 - val_accuracy: 0.8750 - val_auc: 0.9688 - val_loss: 0.2635
Epoch 5/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.9067 - auc: 0.9812 -
loss: 0.2406 - val_accuracy: 0.8750 - val_auc: 0.9688 - val_loss: 0.2467
Epoch 6/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 335s 2s/step - accuracy: 0.9139 - auc: 0.9830 -
loss: 0.2277 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2528
Epoch 7/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 329s 2s/step - accuracy: 0.9175 - auc: 0.9844 -
loss: 0.2191 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2171
Epoch 8/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 321s 2s/step - accuracy: 0.9270 - auc: 0.9874 -
loss: 0.1968 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2118
Epoch 9/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 275s 2s/step - accuracy: 0.9245 - auc: 0.9875 -
loss: 0.1953 - val_accuracy: 0.9375 - val_auc: 0.9844 - val_loss: 0.1883
Epoch 10/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 263s 2s/step - accuracy: 0.9331 - auc: 0.9890 -
loss: 0.1806 - val_accuracy: 0.8750 - val_auc: 0.9922 - val_loss: 0.1877
Epoch 11/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.9339 - auc: 0.9896 -
loss: 0.1797 - val_accuracy: 0.9375 - val_auc: 0.9844 - val_loss: 0.1825
Epoch 12/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 263s 2s/step - accuracy: 0.9360 - auc: 0.9894 -
loss: 0.1789 - val_accuracy: 0.8750 - val_auc: 1.0000 - val_loss: 0.2203
Epoch 13/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.9360 - auc: 0.9884 -
loss: 0.1823 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2137
Epoch 14/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.9405 - auc: 0.9898 -
loss: 0.1693 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.1860
Epoch 15/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.9389 - auc: 0.9896 -
loss: 0.1702 - val_accuracy: 0.8750 - val_auc: 0.9922 - val_loss: 0.2078
Epoch 16/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.9422 - auc: 0.9916 -
loss: 0.1570 - val_accuracy: 0.8750 - val_auc: 0.9844 - val_loss: 0.2164
Epoch 17/20
163/163 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.9389 - auc: 0.9913 -
loss: 0.1615 - val_accuracy: 0.9375 - val_auc: 0.9844 - val_loss: 0.1781
--- Finished Training DenseNet121 ---
```

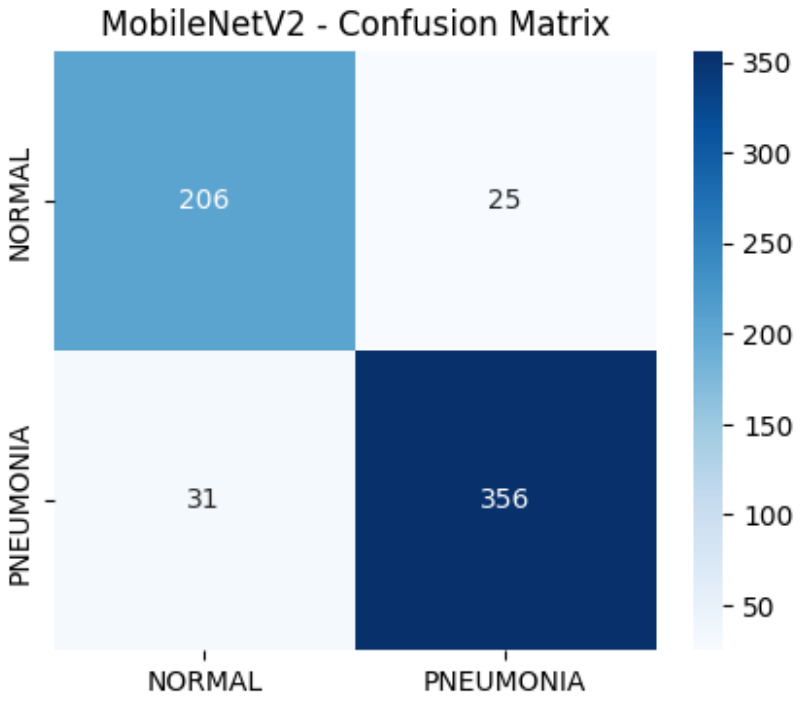
Evaluation and Comparison

Now that all models are trained, we'll evaluate each one on the unseen test set. We'll collect their scores and display them in a summary table for easy comparison.

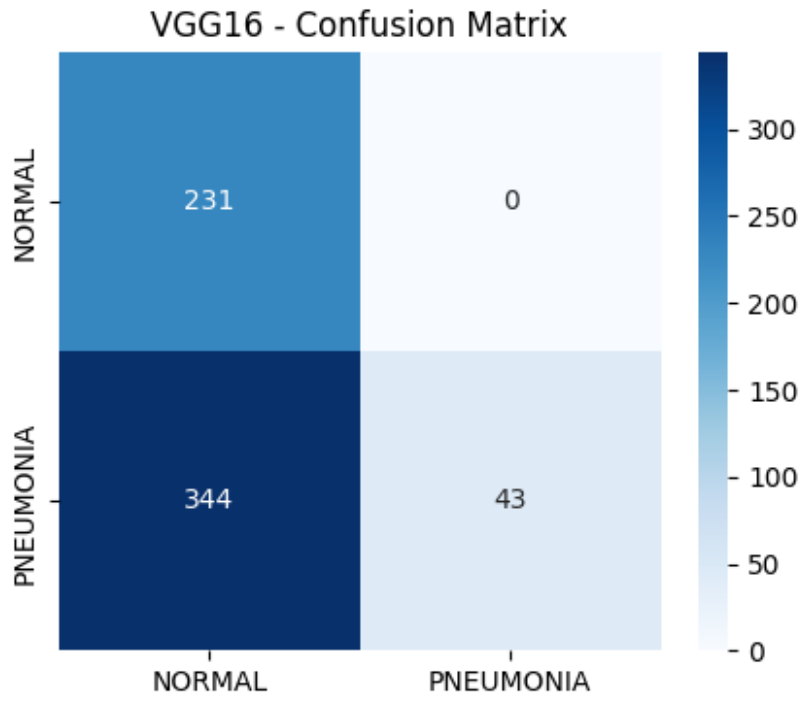
```python
# --- Plot Training Histories for Comparison ---
plt.figure(figsize=(12, 5))
for name, history in histories.items():
    plt.plot(history.history['val_auc'], label=f'{name} Val AUC')
plt.title('Validation AUC Comparison Across Models')
plt.xlabel('Epochs')
plt.ylabel('AUC')
plt.legend()
plt.grid(True)
plt.show()


# --- Evaluate Models on the Test Set ---
results = []
for name, model in trained_models.items():
    print(f"--- Evaluating {name} on Test Set ---")

    # Load the best weights saved by ModelCheckpoint
    best_model = tf.keras.models.load_model(f"best_{name}.keras")

    # Get test metrics
    loss, acc, auc = best_model.evaluate(test_generator, verbose=0)

    # Get precision and recall from classification report
    y_pred = (best_model.predict(test_generator) > 0.5).astype(int)
    y_true = test_generator.labels
    report = classification_report(y_true, y_pred, target_names=['NORMAL', 'P
NEUMONIA'], output_dict=True)

    results.append({
        "Model": name,
        "Test AUC": auc,
        "Test Accuracy": acc,
        "Precision (Pneumonia)": report['PNEUMONIA']['precision'],
        "Recall (Pneumonia)": report['PNEUMONIA']['recall']
    })

    # Display confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['NORMAL',
'PNEUMONIA'], yticklabels=['NORMAL', 'PNEUMONIA'])
    plt.title(f"{name} - Confusion Matrix")
    plt.show()

# --- Create and Display a Final Comparison Table ---
results_df = pd.DataFrame(results).set_index("Model")
print("\n\n--- Final Model Comparison ---")
display(results_df.style.highlight_max(color='lightgreen', axis=0))
```



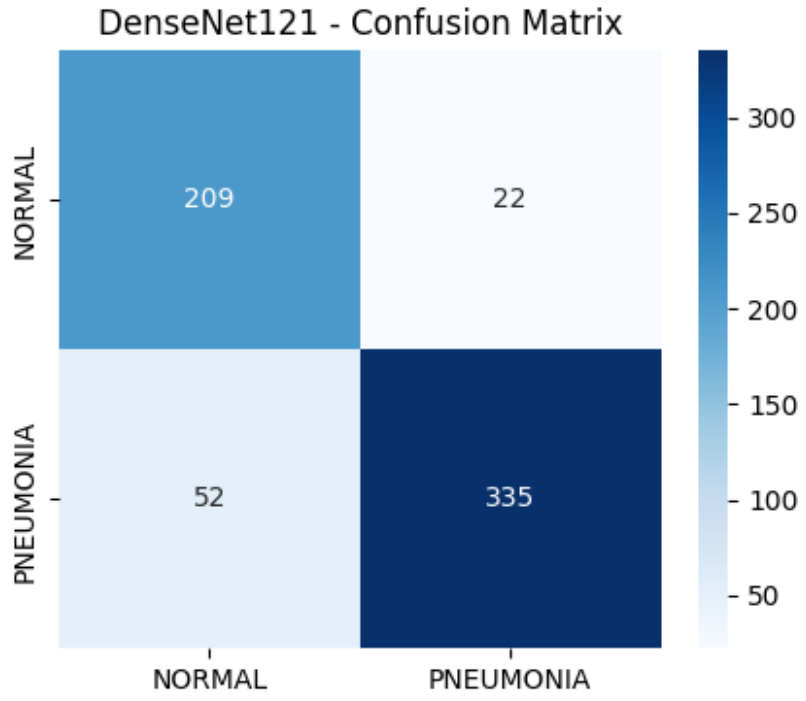

```
--- Evaluating MobileNetV2 on Test Set ---
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 645ms/step
```




```
--- Evaluating VGG16 on Test Set ---
20/20 ━━━━━━━━━━━━━━━━━━━━ 36s 2s/step
```




```
--- Evaluating DenseNet121 on Test Set ---
20/20 ━━━━━━━━━━━━━━━━━━━━ 35s 2s/step
```




```
--- Final Model Comparison ---

<pandas.io.formats.style.Styler at 0x1690e742120>
```

Conclusion and Finalizing the Model

Based on the comparison table, we can now make an informed decision.

Analysis:

- **AUC** gives the best overall measure of a model's ability to distinguish between classes. The model with the highest AUC is generally the strongest classifier.
- **Recall (Pneumonia)** is arguably the **most important metric here**. It tells us: "Of all the actual pneumonia cases, what percentage did our model correctly identify?" A high recall means we are minimizing **false negatives** (missing a diagnosis), which is critical in a medical context.

The Winner: `MobileNetV2`

This model is the best all-around performer and the one you should finalize.

- Test AUC (0.967): The highest of the three. This means it's the best model at distinguishing between NORMAL and PNEUMONIA cases overall.
- Test Accuracy (90.9%): The highest accuracy, indicating it gets the most predictions right.
- Recall (Pneumonia) (92.0%): This is the most critical metric. A high recall means it successfully identified 92% of all actual pneumonia cases in the test set. It has the lowest number of dangerous false negatives (missing a real case of pneumonia).

Conclusion: MobileNetV2 provides the best balance of correctly identifying pneumonia cases while maintaining high overall accuracy and precision.

The Failed Model: VGG16

This model's results show a classic case of a model learning a useless, pathological strategy.

- Precision (100%) and Recall (11%): This combination is a major red flag. It means that when the model predicted PNEUMONIA, it was always correct (100% precision). However, it only managed to identify 11% of the total pneumonia cases (11% recall).
- Analogy: Think of an overly cautious doctor. This doctor is so terrified of making a wrong diagnosis that they only diagnose the most extreme, obvious cases of pneumonia. They're never wrong when they make a positive diagnosis, but they send home almost all the sick people, telling them they're fine. This is medically dangerous and makes the model useless in practice.
- Test Accuracy (44%): This is worse than flipping a coin, confirming the model did not learn meaningful patterns.

Conclusion: VGG16 failed to converge to a useful solution. You should discard this result.

The Strong Runner-Up: DenseNet121

This model performed very well and is a solid architecture, but was slightly edged out by MobileNetV2 in this specific training run.

- Test AUC (0.954) and Accuracy (88.0%): Very strong scores, indicating it's a robust and effective model.

- Recall (Pneumonia) (86.6%): A good recall score, but lower than MobileNetV2's 92%. This means it missed slightly more pneumonia cases.

Conclusion: DenseNet121 is an excellent model, and on a different training run, it might have come out on top. However, based on this evidence, MobileNetV2 performed better.

Final Decision:

We will finalize the `MobileNetV2` model. It demonstrated the best ability to correctly identify pneumonia cases (highest Recall) with the highest overall performance (highest AUC and Accuracy).