****Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Курсова робота

з дисципліни «Системи і методи підтримки прийняття рішень» на тему

«ІСППР для розв’язання задач класифікації зображень»

**Виконав:** студент I курсу

групи КА-42мп

Богдан Ярінко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис)

Перевірив:

Ігор Голінко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оцінка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис)

Київ – 2024

Зміст

[Вступ 4](#_Toc186113916)

[1 Технічне завдання 6](#_Toc186113917)

[1.1 Загальні відомості 6](#_Toc186113918)

[1.2 Призначення та мета створення системи 6](#_Toc186113919)

[1.3 Характеристики об’єкта 6](#_Toc186113920)

[1.4 Вимоги до системи 7](#_Toc186113921)

[1.4.1 Вимоги до системи в цілому 7](#_Toc186113922)

[1.4.2 Вимоги до функцій, які виконуватиме система 7](#_Toc186113923)

[1.4.3 Вимоги до видів забезпечення 8](#_Toc186113924)

[1.5 Склад і зміст робіт із створення системи 9](#_Toc186113925)

[1.6 Порядок контролю і приймання системи 10](#_Toc186113926)

[2 Опис системного підходу до проектування ІСППР 12](#_Toc186113927)

[2.1 Загальні принципи системного підходу 12](#_Toc186113928)

[2.2 Підготовка та обробка даних 13](#_Toc186113929)

[2.3 Ідентифікація та управління невизначеностями 14](#_Toc186113930)

[2.4 Оптимізація моделі та адаптація 15](#_Toc186113931)

[2.5 Альтернативні методи оцінювання 16](#_Toc186113932)

[2.6 Вибір підходів для класифікації зображень 17](#_Toc186113933)

[3 Опис архітектури ІСППР 20](#_Toc186113934)

[3.1 Структурна схема ІСППР 20](#_Toc186113935)

[3.2 Типи запитів до системи 23](#_Toc186113936)

[3.3 Функції системи обробки даних та генерування результатів 24](#_Toc186113937)

[4 Опис вимог до інтерфейсу користувача 27](#_Toc186113938)

[4.1 Загальні вимоги до інтерфейсу 27](#_Toc186113939)

[4.2 Типи інтерфейсів, що використовуються в системі 28](#_Toc186113940)

[4.3 Вимоги до функцій та структурних елементів інтерфейсу 29](#_Toc186113941)

[5 Опис програмної реалізації ІСППР 31](#_Toc186113942)

[5.1 Обгрунтування вибору інструментальної платформи 31](#_Toc186113943)

[5.2 Опис основних класів, функцій та модулів у СППР 32](#_Toc186113944)

[5.3 Інструкція з експлуатації ІСППР 35](#_Toc186113945)

[6 Приклади застосування 37](#_Toc186113946)

[Висновок 45](#_Toc186113947)

[Перелік посилань 47](#_Toc186113948)

[Додатки 48](#_Toc186113949)

# Вступ

У сучасному світі швидкого технологічного розвитку та постійного вдосконалення систем штучного інтелекту особливої актуальності набувають інформаційні системи підтримки прийняття рішень (ІСППР), здатні автоматизувати складні процеси аналізу та класифікації даних. Зокрема, надзвичайно важливим напрямком є автоматична класифікація зображень з використанням сучасних архітектур нейронних мереж та комп'ютерного зору.

Особливої уваги заслуговує розроблення систем класифікації зображень військової техніки, зокрема літаків, що має критичне значення для забезпечення національної безпеки та ефективності стратегічного планування. Традиційні методи ручної ідентифікації та класифікації є надто повільними, трудомісткими та схильними до помилок, що зумовлює нагальну потребу у створенні автоматизованих інтелектуальних систем.

Основні завдання дослідження включають:

1. аналіз існуючих методів та підходів до класифікації зображень;
2. розроблення модульної архітектури інформаційної системи з гнучкою структурою;
3. реалізацію підсистем завантаження, обробки та класифікації зображень;
4. створення зручного користувацького інтерфейсу для ефективної роботи з системою;
5. тестування та оцінка точності розробленої системи на реальних датасетах.

У контексті сучасних викликів у сфері безпеки та технологічного розвитку особливого значення набуває вдосконалення методів аналізу та класифікації зображень військової техніки. Розроблена інформаційна система демонструє перспективні підходи до використання штучного інтелекту для автоматизації процесів ідентифікації військових літаків, що дозволяє підвищити ефективність моніторингу та аналізу складних технічних об'єктів.

Практичне значення розробленої системи визначається її здатністю:

* автоматизувати процеси ідентифікації військової техніки;
* зменшити вплив людського фактора на процес класифікації;
* підвищити швидкість та точність аналізу зображень;
* забезпечити можливість адаптації системи під різні завдання та типи зображень.

Розроблена інформаційна система підтримки прийняття рішень може бути корисною в різних галузях, включаючи військову розвідку, моніторинг повітряного простору, стратегічне планування та аналіз військової техніки.

# 1 Технічне завдання

## 1.1 Загальні відомості

Назва системи: інформаційна система підтримки прийняття рішень для автоматичної класифікації зображень (ІСППР-АКЗ)

Умовне позначення системи: ІСППР-АКЗ

Початок виконання роботи: 02.09.2024

Орієнтовний кінець роботи: 01.01.2025

## 1.2 Призначення та мета створення системи

Призначення системи: система призначена для автоматичної класифікації зображень різного походження з використанням нейронних мереж, методів машинного навчання та комп’ютерного зору.

Мета створення системи: розробка універсальної системи підтримки прийняття рішень, яка дозволяє автоматично класифікувати зображення з використанням сучасних архітектур нейронних мереж та алгоритмів машинного навчання, що забезпечує високу точність аналізу та скорочує час на обробку даних. Основна мета полягає у створенні ефективного інструменту для автоматизації процесів ідентифікації та класифікації військових літаків, що дозволить зменшити людський фактор, підвищити швидкість обробки даних та забезпечити більш точні результати. Це, у свою чергу, сприятиме покращенню стратегічного планування та оперативного реагування у військових операціях.

## 1.3 Характеристики об’єкта

Об’єктом дослідження є цифрові зображення військових літаків, отримані в різних умовах та форматах. Вхідні дані включають зображення, отримані з супутникових знімків, аерофотозйомки та інших джерел, що забезпечують різноманітні кути огляду та освітлення. Основна задача полягає в автоматичній класифікації цих зображень за типами літаків, що дозволить підвищити ефективність моніторингу та аналізу військових об'єктів. Зображення повинні мати розмір не менше 224x224 пікселів та бути чіткими, без значних зашумлень, щоб забезпечити точність класифікації. Система повинна підтримувати формати JPEG та PNG адаптуючись до різних умов освітлення та якості зйомки. Основна мета – створення ефективного інструменту для класифікації військових літаків.

## 1.4 Вимоги до системи

### 1.4.1 Вимоги до системи в цілому

Вимоги до структури та функціонування системи:

* модульна архітектура для легкого розширення, обслуговування та інтеграції нових функцій;
* висока продуктивність для забезпечення швидкої обробки зображень і виконання прогнозів;
* можливість інтеграції сторонніх математичних моделей.

Вимоги до інтерфейсу користувача:

* інтуїтивно зрозумілий інтерфейс із простою навігацією;
* підтримка авторизації користувачів для забезпечення доступу до персоналізованих налаштувань і моделей;
* адаптивний дизайн, оптимізований для роботи на різних пристроях (ПК, планшети, смартфони).

Вимоги до продуктивності:

* мінімальний час завантаження системи;
* оптимальне використання апаратних ресурсів для великих обсягів даних;
* швидка обробка користувацьких моделей для прогнозування.

Вимоги до сумісності:

* підтримка популярних операційних систем (Windows, Linux, macOS);
* сумісність із сучасними браузерами (Chrome, Firefox, Safari, Edge).

### 1.4.2 Вимоги до функцій, які виконуватиме система

Підсистема завантаження та обробки даних:

* можливість завантаження користувачами власних моделей та датасетів для індивідуального прогнозування;
* попередня обробка зображень (масштабування, нормалізація, видалення шуму);
* зберігання оброблених даних у базі для подальшого аналізу.

Підсистема ідентифікації та класифікації:

* використання вбудованих моделей нейронних мереж для класифікації зображень;
* підтримка користувацьких моделей для кастомізованих задач класифікації.

Підсистема візуалізації та звітності:

* відображення результатів класифікації у зручному інтерфейсі;
* генерація звітів та можливість експорту даних;
* налаштування графічних відображень для полегшення інтерпретації результатів.

Підсистема авторизації та управління користувачами:

* реєстрація та авторизація користувачів із захистом даних;
* зберігання персоналізованих налаштувань для кожного користувача;
* контроль доступу до моделей та результатів.

Підсистема управління проектом:

* зручний інтерфейс для завантаження моделей та зміни параметрів прогнозування.
* динамічне увімкнення/вимкнення функцій залежно від потреби користувача та задачі.

### 1.4.3 Вимоги до видів забезпечення

Обслуговування програмного забезпечення:

* виправлення помилок та недоліків, виявлених у програмі;
* оптимізація коду для покращення продуктивності системи.

Документація:

* створення технічної документації для користувачів системи, що міститиме інструкції щодо використання та налаштування;
* забезпечення доступності документації для всіх користувачів, щоб полегшити процес навчання та впровадження системи.

## 1.5 Склад і зміст робіт із створення системи

Створення ІСППР передбачає виконання низки етапів, які деталізовані у таблиці 1. До ключових робіт належать вибір теми, визначення технічних вимог, опис системного підходу до проектування, розробка архітектури системи, програмна реалізація та тестування. Встановлені терміни виконання кожного етапу є важливим елементом управління проєктом і спрямовані на забезпечення своєчасного завершення розробки.

Таблиця 1 – Строки виконання робіт із розробки ІСППР-АКЗ

|  |  |  |
| --- | --- | --- |
| **Вид робіт** | **Зміст роботи** | **Терміни виконання** |
| Вибір теми курсової роботи | Визначення теми курсової роботи. Дослідження джерел, вивчення необхідних теоретичних основ, огляд існуючих ІСППР, пошук датасетів. | 02.09.23 - 14.09.23 |
| Вступ, технічне завдання | Написання вступу, технічного завдання: відображення загальних відомостей, мети, характеристик предметної області, вимог до системи, складу і зміст робіт зі створення системи | 15.09.23 - 05.10.23 |
| Опис системного підходу до проектування ІСППР | Опис загальносистемних рішень, визначення послідовності обробки даних, рішень по математичному забезпеченню | 06.10.23 - 19.10.23 |

Продовження таблиці 1

|  |  |  |
| --- | --- | --- |
| Опис архітектури ІСППР і функціональної схеми | Розробка архітектури ІСППР, побудова структурної схеми, опис призначення кожної підсистеми, наведення списку типів запитів, функцій систем обробки даних та генерації результатів | 20.10.23 - 02.11.23 |
| Опис вимог до інтерфейсу користувача | Формулювання та врахування вимог уявного користувача, дослідження існуючих інтерфейсів на основі меню | 03.11.23 - 15.11.23 |
| Опис програмної реалізації ІСППР та приклади застосування.  Висновки | Програмування модулів системи, їх тестування, інтегрування модулів в єдину систему підтримки прийняття рішень, остаточні випробування на датасетах, оформлення пояснювальної записки | 16.11.23 - 30.11.23 |
| Захист КР | Демонстрація роботи програми, її можливостей, особливостей та пояснювальної записки | 01.12.23 - 14.12.23 |

## 1.6 Порядок контролю і приймання системи

Види випробувань системи:

* функціональне тестування - перевірка відповідності системи функціональним вимогам та специфікаціям;
* нефункціональне тестування - оцінка продуктивності, безпеки, сумісності та інших нефункціональних аспектів системи;
* інтеграційне тестування - перевірка взаємодії між різними модулями та підсистемами системи;
* системне тестування - комплексна перевірка системи в цілому на відповідність усім вимогам
* приймальне тестування - перевірка системи користувачами на відповідність їх очікуванням та вимогам.

Обсяг випробувань системи:

* повне покриття функціональних вимог - тестування всіх функціональних можливостей системи;
* тестування на різних етапах розробки - включає юніт-тести, інтеграційні тести, системні тести та приймальні тести;
* використання реальних даних та сценаріїв - для забезпечення реалістичності результатів випробувань.

Методи випробувань системи:

* ручне тестування - виконання тестів вручну для перевірки функціональності та користувацького досвіду;
* стрес-тестування - перевірка стабільності системи при високих навантаженнях та екстремальних умовах;
* регресійне тестування - перевірка системи після внесення змін для забезпечення відсутності нових дефектів;
* юзабіліті-тестування - оцінка зручності та інтуїтивності користувацького інтерфейсу;
* тестування сумісності - перевірка роботи системи на різних операційних системах, браузерах та пристроях.

Планування випробувань:

* визначення цілей та критеріїв успішності випробувань;
* розробка плану тестування та графіку виконання.

Підготовка середовища для тестування:

* налаштування апаратного та програмного забезпечення;
* створення тестових даних та сценаріїв.

Виконання випробувань:

* проведення запланованих тестів відповідно до плану;
* фіксація результатів та виявлених дефектів.

Аналіз результатів:

* оцінка виконання тестів згідно з встановленими критеріями;
* визначення необхідності коригувальних дій.

# 2 Опис системного підходу до проектування ІСППР

## 2.1 Загальні принципи системного підходу

Системний підхід у проектуванні інформаційних систем підтримки прийняття рішень є фундаментальним принципом, який забезпечує ефективну організацію процесів обробки даних та прийняття рішень. Однією з ключових характеристик такого підходу є ієрархічність архітектури, яка дозволяє моделювати систему у вигляді рівнів із чітким розподілом функцій. Це дає змогу не лише оптимізувати роботу системи, але й полегшити її масштабування та адаптацію до нових умов.

Ієрархічний принцип передбачає поділ архітектури на кілька функціональних рівнів, таких як: рівень збирання даних, обробки інформації, прийняття рішень та візуалізації результатів. Кожен рівень виконує специфічні завдання, що мінімізує взаємозалежність компонентів системи. У свою чергу, модульність забезпечує можливість легкої заміни або додавання компонентів без необхідності змін у загальній структурі.

Значення модульності та масштабованості систем підтверджується численними дослідженнями. Наприклад, у роботах [**Помилка! Закладку не визначено.Помилка! Джерело посилання не знайдено.**] та [2] зазначається, що модульна структура дозволяє зменшити складність інтеграції нових функцій, тоді як масштабованість забезпечує адаптацію системи до зростання обсягів даних.

Переваги:

* гнучкість у налаштуванні та розширенні;
* зменшення складності управління;
* підвищення стійкості системи до помилок.

Недоліки:

* ускладнення початкового етапу проектування;
* вимога високої кваліфікації розробників для реалізації модульного підходу.

На основі аналізу переваг системного підходу, обрана для нашої роботи архітектура буде побудована за принципами ієрархічності та модульності, що забезпечить гнучкість і масштабованість ІСППР.

## 2.2 Підготовка та обробка даних

Підготовка та обробка даних є одним із найважливіших етапів при проектуванні систем підтримки прийняття рішень. Якість даних безпосередньо впливає на точність роботи моделі. Зазвичай процес обробки включає масштабування, нормалізацію, очищення даних від шумів і формування навчальних та тестових вибірок.

Нормалізація даних дозволяє зменшити варіацію значень, що особливо важливо при роботі з нейронними мережами, де великі коливання входів можуть уповільнити процес навчання. Попередня обробка також включає видалення зашумлених даних, яке допомагає уникнути спотворення моделі під час навчання.

Формування навчальних і тестових вибірок – важливий етап для оцінки продуктивності моделі. Використання методів крос-валідації дозволяє отримати об’єктивні результати та уникнути перенавчання.

Переваги сучасних методів обробки даних:

* зниження впливу зашумлених даних;
* підвищення точності та стабільності моделей.

Недоліки:

* висока ресурсомісткість обробки великих обсягів даних;
* необхідність обрання оптимальних параметрів нормалізації та очищення.

Для розроблюваної ІСППР буде використано підхід, що включає нормалізацію вхідних даних, видалення шуму з використанням алгоритмів обробки зображень (наприклад, згладжування на основі Гаусового фільтра), а також формування збалансованих навчальних і тестових вибірок із застосуванням крос-валідації.

Нормалізацію даних будемо робити Z-оцінкою, яка є важливим етапом підготовки даних для нейронних мереж. Цей метод дозволяє перетворити дані так, щоб вони мали середнє значення 0 і стандартне відхилення 1. Стандартизація даних є критично важливою, оскільки нейронні мережі чутливі до масштабу вхідних даних. Великі коливання значень можуть уповільнити процес навчання та призвести до погіршення продуктивності моделі.

Стандартизована оцінка обчислюється наступною формулою:

де, *Z* – стандартизоване значення;

X – початкове значення;

– середнє значення вибірки;

– стандартне відхилення вибірки;

Застосування стандартизації в підготовці даних для нейронних мереж дозволяє зменшити вплив зашумлених даних, підвищити точність та стабільність моделей, а також прискорити процес навчання. Це особливо важливо, оскільки нейронні мережі зазвичай вимагають великої кількості даних для навчання, і якість цих даних безпосередньо впливає на результати моделювання.

## 2.3 Ідентифікація та управління невизначеностями

У процесі класифікації зображень однією з ключових проблем є невизначеність, яка може виникати на різних етапах роботи системи. Серед основних джерел невизначеності можна виділити недостатню якість вхідних даних, невідповідність між тренувальними і реальними даними, а також обмежену узагальнюючу здатність моделі. Такі невизначеності негативно впливають на точність класифікації, знижуючи ефективність системи підтримки прийняття рішень.

Наукові дослідження показують, що існують два основні підходи до управління невизначеностями: зниження невазначеності на етапі підготовки даних та використання ймовірнісних методів у процесі прогнозування. Зокрема, нормалізація, видалення шуму і збалансування вибірок дозволяють зменшити вплив вхідних аномалій. Статистичні та ймовірнісні підходи, такі як Bayesian Neural Networks, забезпечують врахування невизначеності на рівні моделі.

Для нашої системи підтримки прийняття рішень ми інтегруємо механізми зниження невизначеності на етапі підготовки даних та оптимізації. Зокрема, буде використано метод аугментації даних для створення додаткових зображень на основі початкових, що дозволить підвищити стійкість моделі до зміни умов. Крім того, для мінімізації модельної невизначеності застосовуватиметься підхід ансамблювання моделей, який забезпечує стабільність і підвищує точність прогнозів.

Застосування цих методів дозволить підвищити точність класифікації зображень та покращити адаптивність системи до різних умов роботи.

## 2.4 Оптимізація моделі та адаптація

Оптимізація параметрів моделі є одним із ключових етапів при розробці систем підтримки прийняття рішень. Від якості оптимізації залежить здатність моделі точно класифікувати зображення, уникаючи перенавчання або недонавчання. Одним із найпоширеніших підходів є використання градієнтних методів оптимізації, таких як Adam, які забезпечують швидку та стабільну конвергенцію. Важливим аспектом також є регуляризація, яка знижує складність моделі, зберігаючи її здатність узагальнювати.

Крім того, важливо враховувати необхідність динамічної адаптації моделі під час роботи з різними типами даних. Дослідження у цій галузі пропонують підходи, які дозволяють моделі автоматично налаштовувати свою структуру, наприклад, змінювати кількість шарів чи нейронів залежно від характеристик вхідних даних.

У рамках нашої роботи ми плануємо використовувати багато різних оптимізаторів таких як: Adam, різні види градієнтних спусків, генетичні оптимізатори для ефективного налаштування вагових коефіцієнтів, а також регуляризацію L2 та Dropout для зменшення ризику перенавчання. Для забезпечення адаптивності моделі передбачається автоматичне налаштування структури мережі, що дозволить адаптуватися до змінних характеристик зображень.

Для запобігання перенавчанню ми інтегруємо механізм раннього зупинення навчання. Цей підхід дозволить припиняти тренування моделі в момент досягнення оптимальної продуктивності на валідаційній вибірці. У сукупності ці методи забезпечать високий рівень точності та стійкості розроблюваної ІСППР.

## 2.5 Альтернативні методи оцінювання

Ефективність систем підтримки прийняття рішень значною мірою залежить від методів оцінювання її продуктивності. Основними підходами є порівняльний аналіз архітектур, використання ансамблевих методів і комбінування прогнозів. Ці підходи спрямовані на підвищення якості класифікації та зменшення невизначеностей.

Порівняльний аналіз архітектур. Сучасні архітектури нейронних мереж, такі як ResNet, EfficientNet і DenseNet, мають унікальні особливості, які впливають на точність і швидкість класифікації. Порівняння архітектур передбачає оцінку продуктивності моделей на однаковому наборі даних із використанням метрик, таких як точність, повнота і F1-міра. Наприклад, ResNet демонструє високу точність завдяки залишковим зв'язкам, що сприяють уникненню зникання градієнтів, а EfficientNet оптимізований для обчислювальної ефективності.

У нашій роботі буде проведено порівняння кількох архітектур для визначення найкращого балансу між точністю і швидкістю класифікації зображень.

Методи ансамблювання моделей. Ансамблювання – це техніка, яка комбінує результати кількох моделей для покращення точності та стабільності прогнозів. Найбільш популярними підходами є:

* bagging (Bootstrap Aggregating): створення кількох моделей на основі різних підвибірок даних, як у випадку Random Forest;
* boosting: покращення результатів шляхом навчання моделей на помилках попередніх, наприклад, алгоритм XGBoost;
* stacking**:** комбінування прогнозів кількох моделей за допомогою метамоделі.

Для нашої системи буде застосовано методи ансамблювання на їхніх основах, що дозволить враховувати сильні сторони різних моделей і підвищити точність класифікації.

Комбінування прогнозів.Цей підхід включає об'єднання результатів кількох моделей, наприклад, шляхом усереднення або більш складних підходів, таких як голосування або вагове комбінування. Використання вагового підходу дозволяє врахувати надійність кожної моделі, що особливо корисно при роботі з неоднорідними даними.

У рамках нашої роботи буде реалізовано вагове комбінування прогнозів із урахуванням точності кожної моделі на тестових даних. Це дозволить отримати більш стабільний і точний результат.

## 2.6 Вибір підходів для класифікації зображень

Розробка систем класифікації зображень вимагає обґрунтованого вибору методів, які забезпечують оптимальне співвідношення між точністю, швидкістю та ресурсомісткістю.

Аналіз сучасних підходів. Сучасні підходи до класифікації зображень можна розділити на дві основні категорії: традиційні методи обробки зображень та методи на основі машинного і глибокого навчання. Кожен із цих підходів має свої особливості, які визначають їх доцільність у залежності від характеру даних та складності завдання.

Традиційні методи обробки зображень.Традиційні методи обробки зображень базуються на виділенні ключових характеристик (фіч) зображення та подальшому їх використанні у стандартних алгоритмах класифікації. Найбільш поширеними підходами є:

* SIFT (Scale-Invariant Feature Transform): метод виділення особливостей, стійких до масштабування та обертання, який ефективно працює з простими структурами зображень [**Помилка! Джерело посилання не знайдено.**];
* HOG (Histogram of Oriented Gradients): метод аналізу локальних градієнтів і їх орієнтацій, що забезпечує високу точність у задачах розпізнавання об'єктів [**Помилка! Джерело посилання не знайдено.**];
* SVM (Support Vector Machine): класифікатор, що працює на основі розділення даних у багатовимірному просторі з максимальною відстанню між класами [**Помилка! Джерело посилання не знайдено.**].

Традиційні методи ефективні у задачах з обмеженою складністю та невеликим обсягом даних. Проте зі збільшенням розмірів та варіативності датасетів їх продуктивність значно знижується. Основними недоліками цих підходів є складність виділення релевантних ознак вручну та обмежена здатність масштабуватися на складніші задачі.

Методи на основі машинного навчання. Методи машинного навчання, зокрема глибокі нейронні мережі, стали революційним кроком у класифікації зображень. На відміну від традиційних підходів, нейронні мережі автоматично виділяють ключові ознаки зображень, навчаючись безпосередньо на даних. Особливе місце серед них займають Convolutional Neural Networks (CNN), які спеціалізуються на обробці просторових даних, таких як зображення.

Глибокі нейронні мережі мають кілька переваг [**Помилка! Джерело посилання не знайдено.**]:

* автоматизація: вони автоматично знаходять релевантні особливості, зменшуючи потребу в ручній інженерії ознак;
* гнучкість: здатність працювати з різноманітними даними, включаючи зображення, що містять складні структури;
* масштабованість: можливість роботи з великими датасетами та адаптація до складних задач.

Серед сучасних архітектур CNN, що використовуються для класифікації зображень, можна виділити: ResNet, EfficientNet, InceptionNet, MobileNet:

Обґрунтування вибраного методу.Для класифікації зображень у нашій роботі буде проведено порівняння кількох сучасних архітектур глибоких нейронних мереж на основі CNN. Такий підхід обрано через здатність цих моделей автоматично виділяти важливі характеристики зображень, що усуває потребу у складній ручній інженерії ознак. Серед переваг CNN також слід виділити високу продуктивність у задачах класифікації зображень завдяки глибоким шарам і механізмам згортки.

Основна мета дослідження полягає у визначенні архітектури, яка найкраще відповідає вимогам точності, швидкості та ресурсомісткості для задач класифікації зображень військових літаків. У ході роботи буде протестовано такі популярні архітектури, як ResNet, EfficientNet, DenseNet, MobileNet та інші. Кожна з них має унікальні переваги, які можуть бути корисними в різних аспектах функціонування системи підтримки прийняття рішень (ІСППР):

* ResNet забезпечує ефективне навчання завдяки використанню залишкових зв'язків, що дозволяє уникнути проблеми зникнення градієнта в глибоких мережах [**Помилка! Джерело посилання не знайдено.**];
* EfficientNet оптимізована для використання ресурсів, що важливо для роботи системи в умовах обмежених обчислювальних можливостей [**Помилка! Джерело посилання не знайдено.**];
* DenseNet підвищує ефективність навчання за рахунок щільного з'єднання між шарами, що сприяє повторному використанню ознак [**Помилка! Джерело посилання не знайдено.**];
* MobileNet адаптована для роботи на пристроях із низькою обчислювальною потужністю, забезпечуючи збереження точності [**Помилка! Джерело посилання не знайдено.**].

Нейронні мережі, особливо глибокі згорткові архітектури, є найкращим вибором для задач класифікації зображень завдяки їх здатності до обробки великих обсягів даних і адаптації до складних шаблонів. Масштабованість CNN дозволяє легко адаптувати їх до змінних умов роботи, таких як різні формати зображень, кути огляду та освітлення.

Результати порівняльного аналізу допоможуть визначити архітектуру, яка найбільш підходить для реалізації ІСППР у нашій роботі. Це забезпечить високу точність класифікації при збереженні прийнятного рівня ресурсомісткості, що є критично важливим для практичного впровадження системи в реальних умовах.

Таким чином, використання та порівняння сучасних архітектур CNN дозволить досягти високої ефективності роботи системи, що відповідає цілям і вимогам розроблюваної ІСППР.

# 3 Опис архітектури ІСППР

## 3.1 Структурна схема ІСППР

Архітектура ІСППР-АКЗ для класифікації зображень будується за модульним принципом з урахуванням ієрархічної структури. Такий підхід дозволяє забезпечити високу продуктивність, гнучкість і масштабованість системи, що є критично важливими аспектами в умовах швидко змінюваного інформаційного середовища. ІСППР складається з кількох підсистем, які взаємодіють між собою для обробки даних, виконання класифікації та представлення результатів, що забезпечує комплексний підхід до вирішення задачі.

Загальна структурна схема системи представлена на рисунку 3.1 та включає наступні компоненти, кожен з яких виконує свою специфічну функцію в рамках загальної архітектури.

1. Підсистема авторизації

Підсистема авторизації забезпечує ідентифікацію користувачів та управління їх доступом до функціоналу системи. Вона реалізує механізми, які дозволяють перевірити повноваження користувачів, а також створювати й підтримувати облікові записи. Ця підсистема є важливим елементом для забезпечення безпеки інформації, оскільки обмежує несанкціонований доступ до системи, що, в свою чергу, запобігає можливим загрозам і витокам конфіденційних даних. Завдяки надійній системі авторизації, користувачі можуть бути впевнені в тому, що їхні дані захищені, а доступ до критично важливих функцій системи контролюється.

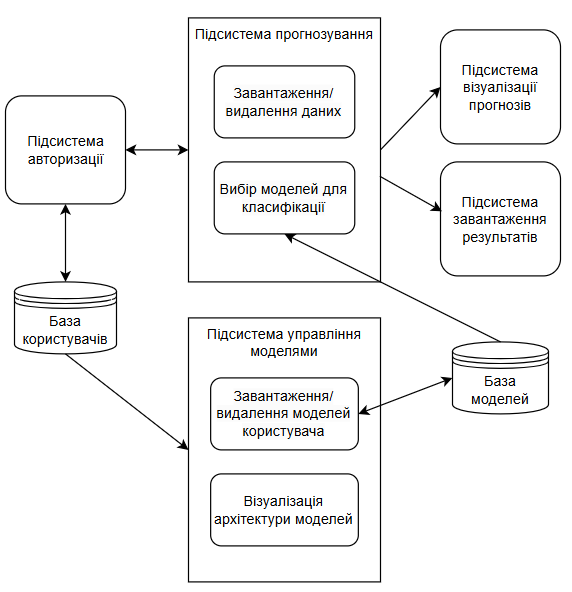


Рисунок 3.1 – Архітектура ІСППР

1. Підсистема прогнозування

Підсистема прогнозування є центральним компонентом ІСППР та відповідає за основні обчислювальні процеси. Вона включає:

* завантаження/видалення даних, що забезпечує прийом вхідної інформації від користувача або зовнішніх джерел і підготовку даних для подальшої обробки;
* вибір моделей для класифікації, що дозволяє користувачеві обирати оптимальну модель для виконання задачі класифікації. У ІСППР буде наведено приклади використання таких моделей, як ResNet, EfficientNet, MobileNet, Це забезпечує гнучкість системи у використанні різних моделей глибокого навчання залежно від специфіки даних.

Ця підсистема інтегрується з іншими модулями для збереження проміжних і кінцевих результатів.

1. Підсистема управління моделями

Підсистема управління моделями відповідає за роботу з модельним пулом, зокрема:

* завантаження або видалення моделей користувача;
* візуалізацію архітектури моделей, що дозволяє користувачам переглядати структуру завантажених або використовуваних моделей.

Ця підсистема інтегрується з базою моделей, де зберігаються всі актуальні архітектури, завантажені користувачами чи системою.

1. База користувачів та база моделей

* база користувачів містить інформацію про зареєстрованих користувачів та їх права доступу;
* база моделей зберігає всі наявні моделі, які можуть бути використані для класифікації, а також їх параметри та метрики продуктивності.

1. Підсистема завантаження результатів

Ця підсистема забезпечує збереження й передачу результатів роботи системи користувачу. Завдяки цьому модулю прогнозовані результати можуть бути завантажені для подальшого аналізу або інтегровані в інші системи.

1. Підсистема візуалізації прогнозів

Підсистема візуалізації прогнозів надає графічне представлення результатів класифікації для кращого сприйняття користувачем. Вона інтегрується з підсистемою прогнозування та результатів, отримуючи дані для побудови візуалізацій.

Підсистеми тісно взаємодіють одна з одною завдяки централізованому обміну даними через бази користувачів та моделей. Підсистема авторизації забезпечує контроль доступу до функціоналу, а підсистема управління моделями інтегрується з прогнозуванням для використання актуальних архітектур. Результати роботи моделі передаються до підсистеми завантаження результатів та візуалізації, що забезпечує їх подальше представлення користувачу.

Таким чином, розроблена архітектура є ефективною та взаємодіючою системою, що забезпечує повний цикл класифікації зображень: від завантаження даних до представлення результатів.

## 3.2 Типи запитів до системи

У контексті розробки інформаційної системи підтримки прийняття рішень для класифікації зображень, важливим аспектом є розробка ефективного інтерфейсу користувача, здатного забезпечити інтерактивну взаємодію з системою. Це передбачає реалізацію різноманітних типів запитів, які дозволять користувачам виконувати широкий спектр операцій, від завантаження даних до аналізу результатів. Запити, які буде реалізовано:

* запит на завантаження даних та моделей; система повинна забезпечити користувачам можливість завантажувати зображення, моделі нейронних мереж та датасети, а також оновлювати або доповнювати наявні моделі, проводячи перевірку цілісності та якості завантажених даних для забезпечення коректності подальшої обробки;
* запит на вибір моделі для класифікації; система має надавати можливість вибору кастомізованих моделей, що дозволить користувачам адаптувати систему до конкретних потреб їхніх задач;
* запит на класифікацію зображень; система має забезпечити можливість обчислення результатів класифікації з використанням вибраних моделей, запускати паралельну обробку зображень для підвищення продуктивності, а також отримувати проміжні результати для відстеження процесу класифікації;
* запит на візуалізацію результатів; система повинна забезпечувати відображення результатів класифікації у вигляді таблиць, графіків, теплових карт, а також забезпечувати інтерактивну візуалізацію з можливістю аналізу конкретних зображень;
* запит на збереження та експорт результатів; користувачі повинні мати можливість зберігати результати у зручному для них форматі;
* запит на авторизацію та персоналізацію користувача; система повинна забезпечити можливість реєстрації та авторизації користувачів, надання доступу до персоналізованих налаштувань та проектів;
* запит на розширення функціоналу системи; користувачі повинні мати можливість додавати нові моделі нейронних мереж, які вони попередньо натренували;
* запит на ретроспективний перегляд результатів; користувачі повинні мати можливість переглядати та порівнювати результати виконаних класифікацій, повторно використовувати збережені моделі та результати.

## 3.3 Функції системи обробки даних та генерування результатів

Система обробки даних та генерування результатів (СОДГР) в рамках (ІСППР), що розробляється, призначена для виконання функцій, пов'язаних із класифікацією зображень. Основна мета СОДГР полягає у забезпеченні процесу обробки даних – від попередньої обробки завантажених зображень до генерації результатів класифікації. СОДГР буде зосереджена на використанні попередньо навчених моделей для класифікації, без внутрішньої можливості навчання або оцінки цих моделей.

Система дозволятиме користувачам завантажувати як самі зображення, які потрібно класифікувати, так і моделі нейронних мереж, які будуть використовуватися для цієї класифікації. Підтримувані моделі будуть класифікаційними, тобто призначені для визначення належності вхідного зображення до певного класу. Система забезпечить підтримку різних архітектур моделей, що дозволить користувачам вибирати відповідну модель для своїх конкретних задач.

Функції СОДГР включатимуть попередню обробку завантажених зображень для підготовки їх до подальшої обробки моделлю. Цей етап може включати зміну розміру, нормалізацію, видалення шуму або інші операції, залежно від вимог моделі. Після попередньої обробки зображення передається на вхід вибраної моделі.

Функції прогнозування у СОДГР передбачають визначення класу для кожного вхідного зображення з використанням імовірнісного підходу (Softmax), що дозволяє отримати ймовірність належності до кожного класу. Результатом роботи системи буде вивід найімовірнішого класу зображення, а також можливість аналізу альтернативних прогнозів у випадках, коли впевненість моделі є недостатньо високою. Користувач зможе бачити ймовірності приналежності до кожного з класів, що дозволить йому самостійно оцінити якість прогнозу.

У даній системі обробки даних та генерування результатів функції навчання моделей розпізнавання образів не передбачені через низку обґрунтованих причин, які пов’язані зі складністю реалізації та специфікою використання таких моделей. Навчання моделей, зокрема глибоких нейронних мереж, потребує значних обчислювальних ресурсів. Це вимагає наявності потужного обладнання, такого як графічні процесори (GPU) або спеціалізовані пристрої (TPU), які є дорогими та не завжди доступними для кінцевого користувача. Крім того, включення таких обчислень до системи значно ускладнило б її інфраструктуру, що суперечить меті забезпечення зручності та універсальності.

Ще однією вагомою причиною є необхідність великих обсягів маркованих даних для навчання моделей. Створення та забезпечення якісного датасету – складний і витратний процес, який часто виходить за рамки завдань, поставлених перед системою. До того ж моделі, навчання на певному наборі даних, можуть демонструвати гарні результати лише на даних аналогічної природи. Забезпечення їх універсальності вимагає додаткових механізмів адаптації, які ускладнюють роботу системи.

Однак, щоб компенсувати відсутність функцій навчання, СОДГР реалізує механізми, що дозволяють динамічно підключати математичні моделі та параметри вже навчених нейронних мереж. Такий підхід забезпечує гнучкість використання системи. Користувач може інтегрувати власні моделі, попередньо навчені в середовищах, таких як TensorFlow чи PyTorch, що дозволяє адаптувати систему до найсучасніших підходів до розпізнавання образів.

Динамічне підключення вже навчених моделей також значно знижує обчислювальні витрати. Система зосереджує ресурси на етапі прогнозування, уникаючи потреби в навчанні, що робить її придатною для використання на звичайних пристроях, включаючи персональні комп’ютери чи сервери середнього рівня. До того ж, користувачі мають можливість оновлювати або замінювати підключені моделі, забезпечуючи відповідність актуальним задачам без необхідності інтеграції складних механізмів навчання.

Оскільки СОДГР не виконує навчання моделей і не використовує інструментів для оцінки якості моделей за допомогою метрик, таких як точність, повнота, F1-міра, матриця помилок чи ROC-AUC, основний акцент зроблено на забезпеченні стабільності та точності процесу прогнозування. Вибір моделі та оцінка її відповідності до конкретної задачі покладаються виключно на користувача. Для цього система дозволяє інтегрувати параметри навчених моделей у стандартних форматах, що спрощує їх імплементацію. Таким чином, СОДГР надає користувачам необхідний рівень гнучкості, залишаючи складніші задачі навчання поза своєю зоною відповідальності.

Метод прийняття рішення буде базуватися на аналізі прогнозів моделі, які надає система. Якщо, наприклад, система класифікує зображення з низькою впевненістю, користувачеві може бути рекомендовано переглянути зображення вручну або використовувати іншу модель. Таким чином, система надає інформацію для підтримки прийняття рішень, але не гарантує абсолютної точності класифікації.

Отже, основні функції СОДГР у системі класифікації зображень включають попередню обробку завантажених даних, використання завантажених моделей для прогнозування класу зображення, надання ймовірностей приналежності до різних класів та підтримку прийняття рішень на основі цих прогнозів. Система зосереджена на ефективній класифікації з використанням зовнішніх моделей, залишаючи оцінку їх якості та адекватності на розсуд користувача. Це забезпечує гнучкість та можливість застосування різних моделей без необхідності внутрішньої реалізації навчання та оцінки.

# 4 Опис вимог до інтерфейсу користувача

## 4.1 Загальні вимоги до інтерфейсу

Інтерфейс користувача є ключовим компонентом ІСППР, що забезпечує взаємодію між системою та користувачем. Ефективний інтерфейс сприяє швидкому освоєнню системи, дозволяє користувачам інтуїтивно виконувати всі необхідні операції та отримувати необхідні результати. У рамках ІСППР для класифікації зображень інтерфейс повинен бути достатньо гнучким, щоб підтримувати завантаження даних, вибір моделей, виконання класифікації та аналіз результатів.

Розробка інтерфейсу є логічним продовженням етапу створення архітектури ІСППР, описаного в попередньому розділі. Функціонал інтерфейсу має відповідати закладеним функціям системи, забезпечувати швидкий доступ до всіх модулів та підтримувати ефективну взаємодію користувача з системою.

Інтерфейс користувача має відповідати сучасним вимогам зручності, інтуїтивності та ефективності взаємодії. Основні вимоги включають урахування психологічних особливостей користувачів, адаптивність до їхніх потреб, а також дотримання принципів зручності у представленні інформації.

Перша вимога полягає у врахуванні особливостей пам’яті та сприйняття користувачів. Інтерфейс повинен мінімізувати навантаження на короткострокову пам’ять за рахунок розташування важливої інформації у фіксованих зонах екрану та забезпечення її швидкого доступу. Важливо також враховувати час реакції користувачів, забезпечуючи миттєвий відгук системи на виконання запитів.

Другою важливою вимогою є адаптивність інтерфейсу. Система повинна враховувати різні типи даних, що використовуються у задачах класифікації, та адаптувати спосіб їх представлення до потреб користувача. Наприклад, результати класифікації будуть подані у вигляді графіків, таблиць, текстових повідомлень та інтерактивних візуалізацій. Користувач повинен мати змогу налаштовувати типи представлення результатів відповідно до своїх уподобань.

Інтерфейс також має забезпечувати зручну організацію діалогу між користувачем та системою. Буде реалізовано командні віконні форми, меню та стандартизовані форми для введення даних. Важливо, щоб кожен елемент інтерфейсу був логічно розташований і легко доступний для виконання основних функцій, таких як завантаження даних, вибір моделі, запуск класифікації або отримання результатів.

## 4.2 Типи інтерфейсів, що використовуються в системі

У системі для класифікації зображень передбачено використання кількох типів інтерфейсів, які забезпечують зручність взаємодії користувача з системою та ефективність виконання основних функцій.

Основним типом є графічний інтерфейс користувача (GUI), що дозволяє інтуїтивно взаємодіяти із системою за допомогою візуальних елементів, таких як кнопки, меню, випадаючі списки та вікна. Графічний інтерфейс реалізовуватиме такі функції: завантаження даних, вибір моделей класифікації, запуск обчислень і перегляд результатів класифікації. Для зручності користувача інтерфейс розділено на кілька зон, які включають панель управління, зону для візуалізації результатів і область повідомлень системи.

Крім цього, реалізується інтерфейс, заснований на меню, що забезпечує структуровану взаємодію користувача з системою. Користувач має змогу перемикатися між основними розділами (завантаження даних, класифікація, перегляд результатів) за допомогою багаторівневого меню. Такий підхід зручний для покрокового виконання задач і забезпечує логічну організацію роботи з системою.

## 4.3 Вимоги до функцій та структурних елементів інтерфейсу

Інтерфейс ІСППР реалізовує всі необхідні функції для забезпечення зручної взаємодії користувача із системою. Основні вимоги до функцій і структурних елементів інтерфейсу спрямовані на підвищення ефективності роботи користувача з системою класифікації зображень.

Структура інтерфейсу передбачає розділення екрану на кілька основних зон, початкову сторінку можемо побачити на рисунку 4.1.

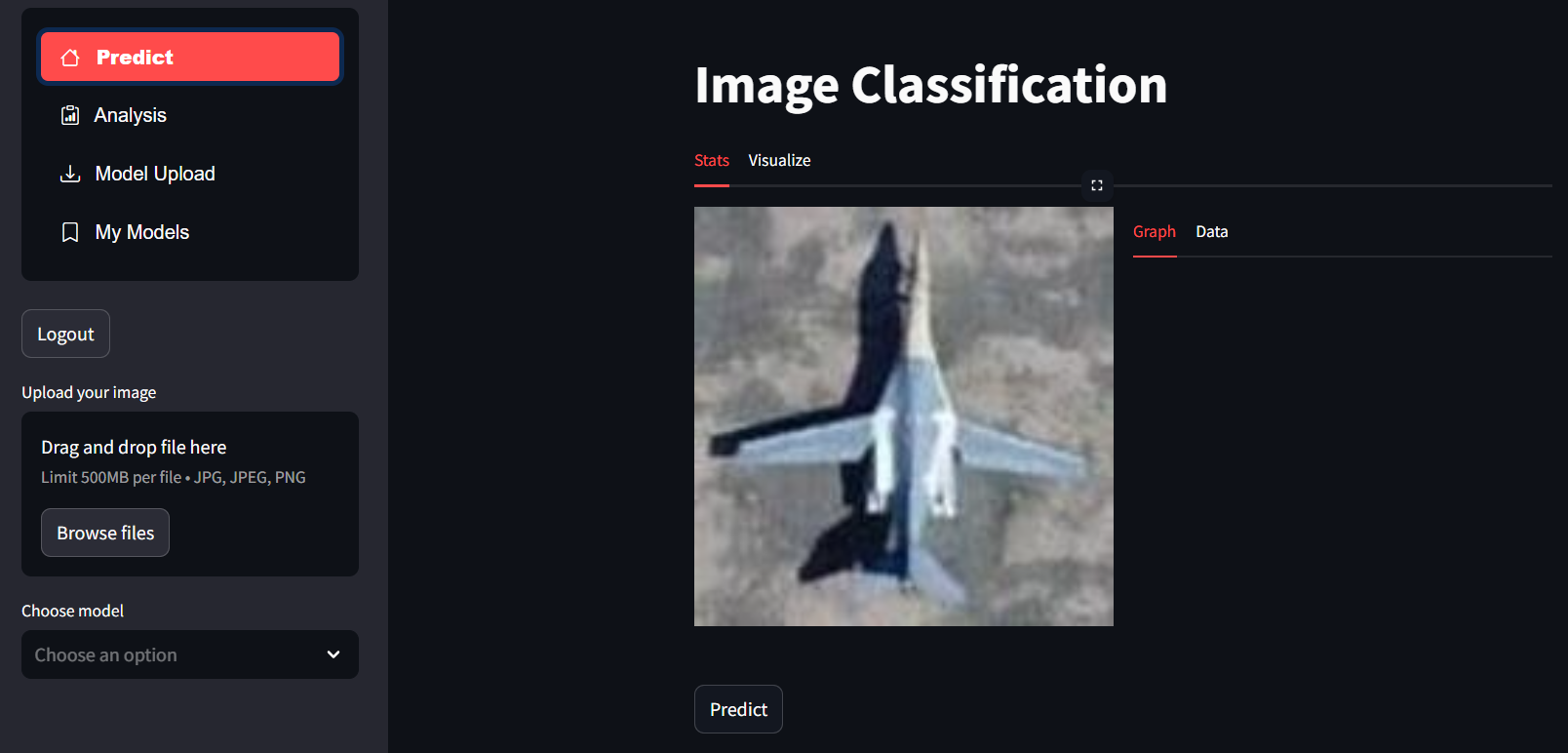


Рисунок 4.1 – Головний інтерфейс ІСППР

Панель управління, розташована у лівій частині екрана, забезпечує доступ до головних функцій системи, таких як завантаження даних, вибір моделі, запуск класифікації та збереження результатів. У центральній частині екрана розташована зона візуалізації, яка відображає результати класифікації у вигляді графіків, таблиць та теплових карт.

Функціональність інтерфейсу включає наступні елементи:

1. механізм завантаження даних: користувач має змогу завантажувати зображення або датасети за допомогою інтуїтивно зрозумілого діалогового вікна із підтримкою різних форматів файлів (наприклад, JPEG, PNG, CSV);
2. вибір моделі: забезпечується інтерактивна панель, де користувач може вибирати власні моделі для прогнозування;
3. перегляд результатів: після виконання класифікації результати відображаються у зоні візуалізації. Користувач може переключатися між різними типами представлення даних (графіки, таблиці, теплові карти);
4. інтерактивність: інтерфейс забезпечує можливість взаємодії з графіками, наприклад, для аналізу окремих класифікованих об’єктів.

Для забезпечення зручності всі елементи інтерфейсу стандартизовані. Інформація, команди та повідомлення завжди з’являються у визначених зонах, що знижує когнітивне навантаження на користувача. Наприклад, результати завжди відображаються в центральній зоні, а повідомлення про помилки – у правій верхній частині, як це показано на рисунку 4.2.

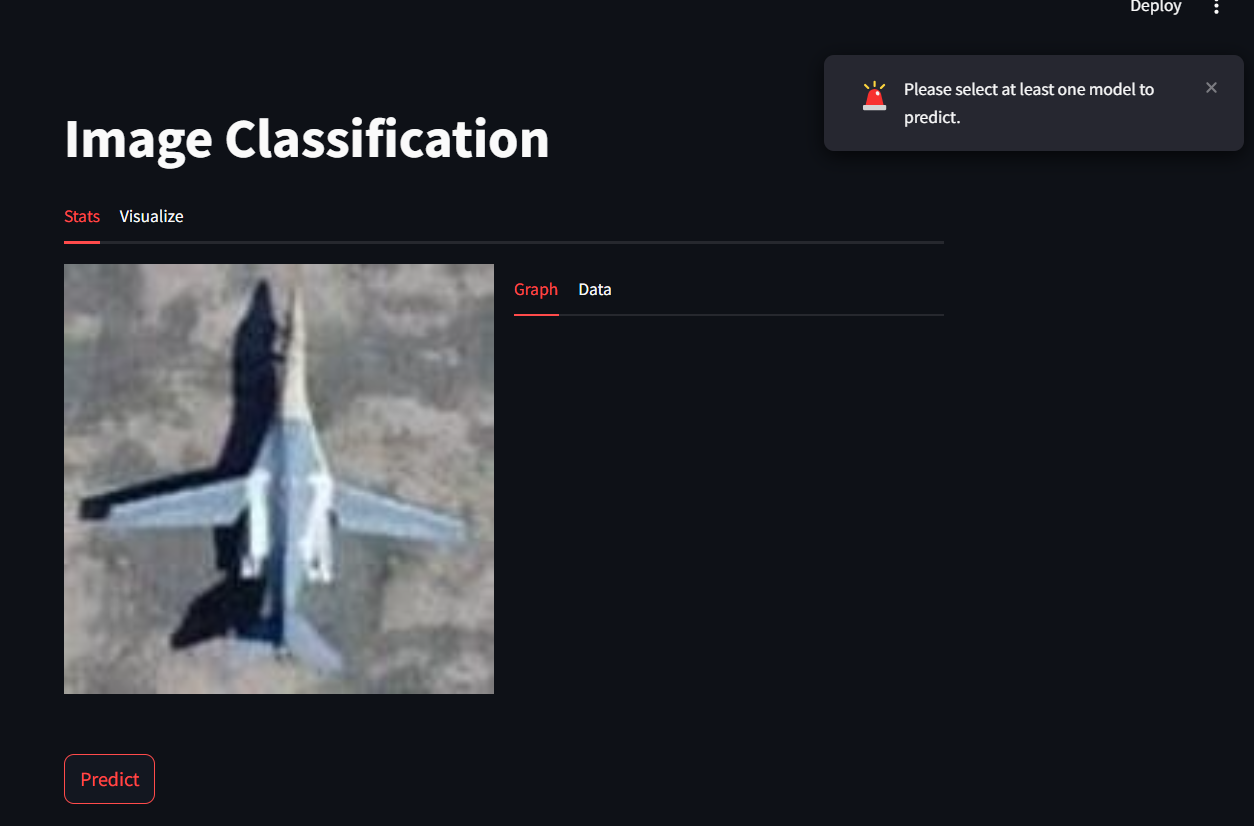


Рисунок 4.2 – Повідомлення про помилку користувачу через те, що він не обрав жодної моделі

Розроблений інтерфейс користувача для ІСППР є ключовим елементом, який забезпечує зручну та інтуїтивну взаємодію користувача із системою. Він дозволяє ефективно виконувати всі необхідні операції, починаючи із завантаження даних і вибору моделей, та завершується візуалізацією результатів класифікації. Графічний інтерфейс системи побудований із урахуванням сучасних стандартів організації візуальних елементів, включаючи чітке розділення екрану на функціональні зони, такі як панель управління, зона виводу результатів та панель повідомлень. Це дозволяє мінімізувати когнітивне навантаження на користувача і спрощує процес аналізу даних.

Таким чином, запропонований інтерфейс є універсальним, гнучким та ефективним рішенням для взаємодії користувача із системою, забезпечуючи виконання всіх закладених функцій та підвищуючи зручність роботи.

# 5 Опис програмної реалізації ІСППР

## 5.1 Обгрунтування вибору інструментальної платформи

Для реалізації даної системи було обрано мову програмування Python та бібліотеку Streamlit як інструментальну платформу для розробки інтерфейсу. Цей вибір обумовлений кількома ключовими факторами.

Python є однією з найпопулярніших мов програмування для наукових досліджень, обробки даних та розробки систем штучного інтелекту. Його простота у використанні та велика кількість бібліотек дозволяють ефективно розв’язувати задачі класифікації зображень, інтегрувати моделі машинного навчання та створювати інтуїтивно зрозумілі інтерфейси.

Streamlit, як інструментальна платформа для створення веб-додатків, забезпечує простий та швидкий спосіб реалізації інтерактивного інтерфейсу. Переваги Streamlit включають:

1. простота використання: інтерфейс створюється за допомогою простих Python-скриптів, що мінімізує час розробки;
2. інтерактивність: підтримка динамічної взаємодії з користувачем, включаючи можливість завантаження файлів, налаштування параметрів та отримання миттєвого відгуку;
3. вбудовані візуалізації: платформа має готові інструменти для відображення графіків, таблиць та інших візуальних елементів. Це дозволяє легко реалізувати зручне представлення результатів класифікації;
4. мінімальні інфраструктурні вимоги: Streamlit дозволяє запускати додаток локально або розгортати його у хмарі без необхідності складної серверної інфраструктури.

Щодо бібліотек для обробки зображень та класифікації, було обрано такі модулі:

* TensorFlow для інтеграції попередньо навчених моделей;
* Pillow для базової обробки зображень (масштабування, нормалізація);
* Numpy та Pandas для роботи з даними та побудови графіків.

Використання Streamlit у поєднанні з Python дозволяє створити повнофункціональну, адаптивну та інтерактивну систему класифікації зображень із мінімальними витратами часу на розробку. Такий підхід забезпечує як гнучкість у реалізації складних функцій, так і зручність користувацького досвіду.

Обрані інструменти також підтримують сучасні стандарти розробки програмного забезпечення, такі як модульність, відкритість коду та можливість легкого розширення функціоналу. Це дозволяє інтегрувати додаткові модулі або модифікувати існуючі без значних зусиль. Таким чином, Python у поєднанні зі Streamlit є оптимальним вибором для реалізації ІСППР.

## 5.2 Опис основних класів, функцій та модулів у СППР

Розроблена система підтримки прийняття рішень для класифікації зображень реалізована у вигляді веб-додатку на базі Streamlit. Архітектура програмного забезпечення побудована з урахуванням модульності, що забезпечує легкість у підтримці, розширенні функціоналу та взаємодії між компонентами. Основні функції програми згруповано в логічні модулі, кожен із яких виконує специфічні завдання у межах загальної системи.

1. Модуль ініціалізації додатку *main.py*

Модуль відповідає за ініціалізацію та запуск Streamlit додатку, управління автентифікацією користувачів, а також маршрутизацію між різними сторінками, його програмний код наведено в Додатоку А. Основними функціями є:

* ініціалізація додатку: запуск Streamlit додатку, налаштування конфігураційних параметрів;
* управління автентифікацією користувачів: вхід користувачів у систему, хешування та перевірка паролів, збереження стану автентифікації у змінній [*st.session\_state*](vscode-file://vscode-app/c:/Users/b.yarinko/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html);
* маршрутизація між сторінками: відображення навігаційного меню, перехід між сторінками додатку: «Predict», «Analysis», «Model Upload», «My Models».

1. Модуль автентифікації користувачів

Модуль забезпечує автентифікацію користувачів, їхній доступ до функціоналу системи та збереження безпеки даних. Знаходиться він у файлі ініціалізації додатку (*main.py*):

* функція *register\_user()* відповідає за реєстрацію нових користувачів, зберігаючи хешовані паролі у таблиці users бази даних PostgreSQL;
* функція *login\_user()* виконує перевірку даних для входу через хешування пароля та порівняння збереженого значення з базою даних;
* таблиця *users* містить інформацію про користувачів: імена, хешовані паролі та права доступу.

1. Модуль управління базою даних *database.py*

Цей модуль забезпечує роботу з базою даних PostgreSQL, яка зберігає інформацію про користувачів, моделі та їхні метадані. Програмний код наведений в Додатку Б.

* функція *get\_connection()* створює підключення до бази даних, використовуючи пул з’єднань для підвищення ефективності;
* CRUD-функції реалізують операції додавання, отримання, оновлення та видалення даних із таблиць *users* та *models*;
* таблиця *models* зберігає інформацію про моделі, зокрема їхні імена, опис, та шлях до файлів.

1. Модуль прогнозування аналізу та візуалізації *predict.py, analysis.py*

Основний модуль, що відповідає за виконання класифікації зображень і аналіз результатів а також їхню візуалізацію, Додаток В та Додаток Г:

* функція *predict()* приймає оброблене зображення та модель, виконує прогнозування й повертає ймовірності для кожного класу;
* функція *generate\_saliency\_map()* створює карту активації, яка візуалізує області зображення, що вплинули на прогноз моделі;
* функція *st.dataframe()* відображає результати прогнозування у вигляді таблиці із зазначенням класів і ймовірностей;
* функція *visualize\_saliency\_map()* візуалізує карту активації для конкретного зображення.

1. Модуль управління моделями *my\_models.py, upload.py*

Цей модуль забезпечує завантаження, збереження, перегляд і видалення моделей машинного навчання. Програмний код наведений в Додатку Д та Додатку Е.

* функція *add\_model()* реалізує завантаження нової моделі у форматі H5. Завантажена модель зберігається у папці *user\_models*, а її метадані додаються до таблиці *models* у базі даних;
* функція *delete\_model()* видаляє модель із файлової системи та бази даних.

Основні змінні системи

* *user\_session*: містить стан користувача, включаючи статус авторизації та вибрані параметри;
* *uploaded\_images*: список зображень, завантажених у поточній сесії;
* *selected\_model*: обрана модель класифікації;
* *classification\_results*: результати прогнозування для кожного завантаженого зображення;
* *db\_connection*: підключення до бази даних для виконання запитів.

Ці функції та змінні забезпечують повну функціональність ІСППР, дозволяючи виконувати класифікацію зображень, керувати моделями та взаємодіяти з користувачами у зрозумілому інтерфейсі.

## 5.3 Інструкція з експлуатації ІСППР

1. Запуск системи

1.1. Переконайтеся, що всі необхідні компоненти встановлено:

* Python (версія 3.8 або вище);
* бібліотеки: Streamlit, TensorFlow, Pandas, Pillow, Numpy, psycopg2;
* PostgreSQL для роботи з базою даних.

1.2. Виконайте наступні команди для запуску системи:

1.2.1. перейдіть до директорії з проєктом:

* *cd path\_to\_project*

1.2.2. запустіть додаток Streamlit:

* *streamlit* run main.py

1.3. Після запуску відкрийте веб-браузер за адресою:

* *http://localhost:8501*

1. Реєстрація та авторизація

2.1. Якщо ви новий користувач, натисніть кнопку «Реєстрація» на головній сторінці.

2.2. Введіть бажаний логін та пароль. Пароль зберігається у хешованому вигляді для забезпечення безпеки.

2.3. Після реєстрації авторизуйтесь, ввівши свої дані у відповідні поля на сторінці входу.

1. Завантаження даних

3.1. Перейдіть до розділу «Завантаження даних».

3.2. Натисніть «Обрати файл» і завантажте зображення.

3.3. Усі завантажені файли будуть збережені в системі та відображені у списку завантажених зображень.

1. Вибір моделі та налаштування параметрів

4.1. Перейдіть до розділу «Моделі».

4.2. Завантажте власну модель через кнопку «Завантажити модель».

4.3. У діалоговому вікні задайте необхідні параметри:

1. Виконання класифікації

5.1. Після завантаження даних і вибору моделі натисніть кнопку «Класифікувати».

5.2. Система виконає класифікацію завантажених зображень, і прогрес буде відображено у панелі повідомлень.

1. Перегляд результатів

6.1. Перейдіть до розділу «Результати».

6.2. Ознайомтеся з результатами класифікації у вигляді таблиці, яка містить:

* назву зображення;
* клас, до якого віднесено зображення;
* ймовірності для кожного класу.

6.3. Відобразіть карту активації (saliency map) для аналізу важливих областей зображення.

1. Експорт результатів

7.1. Натисніть на три крапки у верхньому куті таблиці, щоб зберегти результати у потрібному форматі.

7.2. Оберіть місце збереження файлу на локальному комп’ютері.

1. Управління моделями

8.1. Для завантаження нової моделі натисніть «Завантажити модель» у розділі «Моделі».

8.2. Якщо модель більше не потрібна, видаліть її через кнопку «Видалити модель».

1. Завершення роботи

9.1. Для завершення роботи системи закрийте браузер та зупиніть виконання програми у терміналі через комбінацію клавіш Ctrl+C.

9.2. Переконайтеся, що всі важливі дані експортуються перед виходом.

# 6 Приклади застосування

Розроблена інформаційна система підтримки прийняття рішень для класифікації зображень була протестована на різних прикладах для оцінювання її функціональних можливостей, точності класифікації та здатності до роботи з альтернативними моделями. Наведемо приклади використання системи, включаючи побудову моделей, отримання результатів прогнозів та їх візуалізації.

На рисунку 6.1 зображена головна сторінка системи, яка відкривається користувачу після успішної реєстрації. На ній ми бачимо повідомлення з проханням завантажити своє зображення для виконання прогнозів, та перш ніж це зробити, ми маємо завантажити свою, попередньо навчену нейронну мережу.

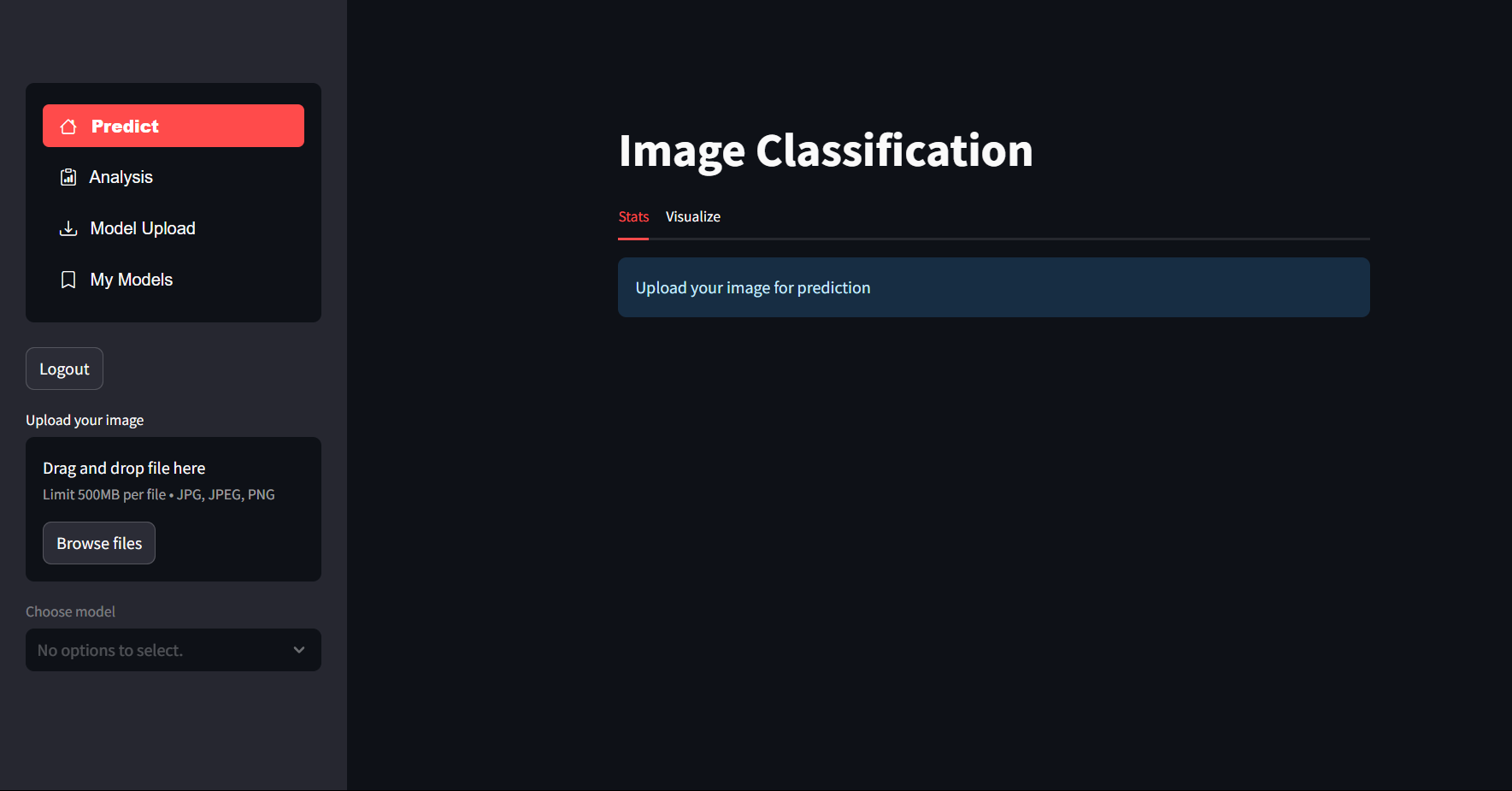


Рисунок 6.1 – Головна сторінка системи

Перейдемо на сторінку завантаження моделей та завантажимо нашу модель (рис. 6.2).

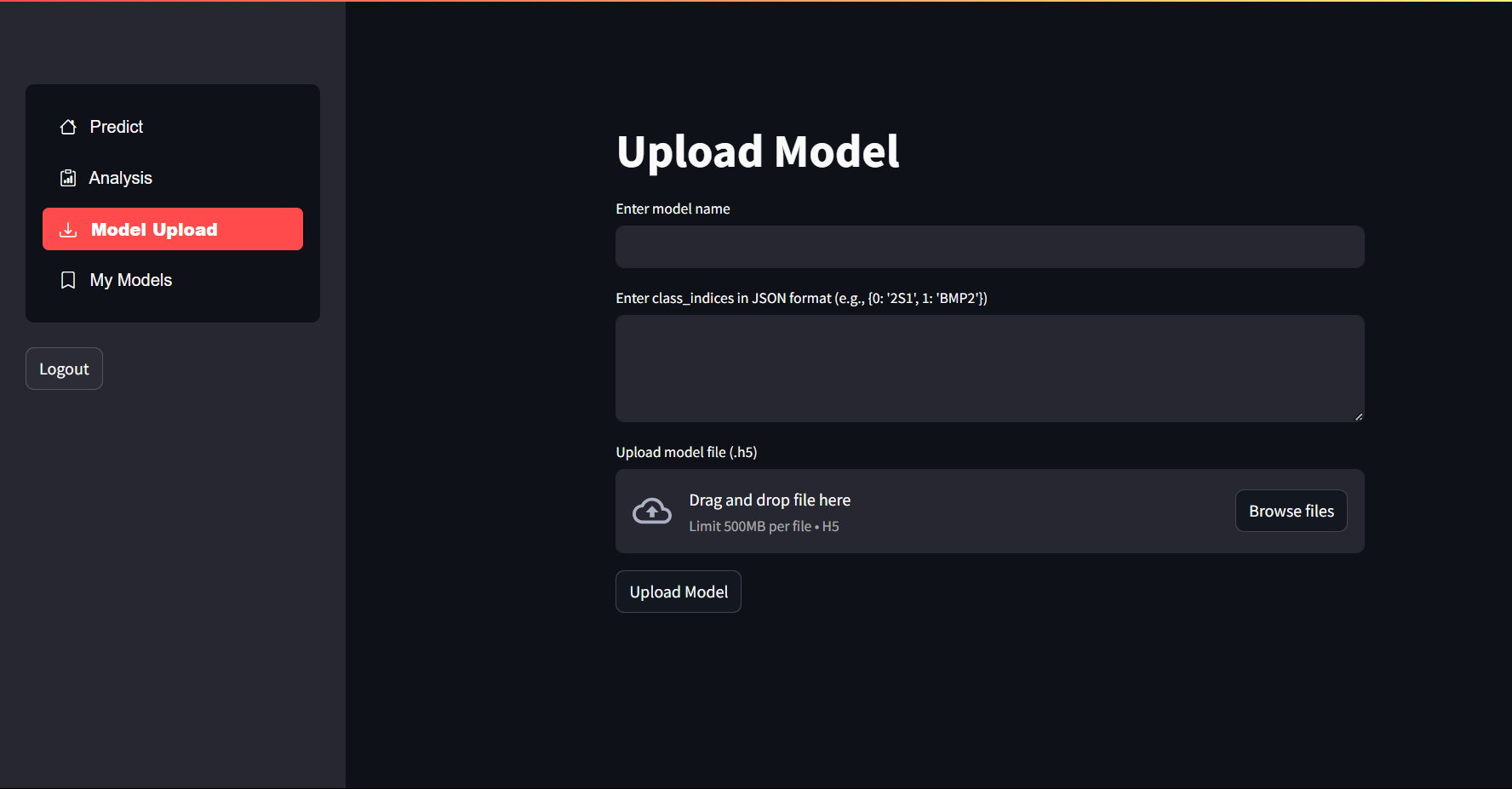


Рисунок 6.2 – Сторінка завантаження моделей

Додамо файл нейронної мережі, назви класів у форматі словника (його можна знайти використавши метод .*class\_indices.keys()* під час навчання моделі) а також назву (рис 6.3).

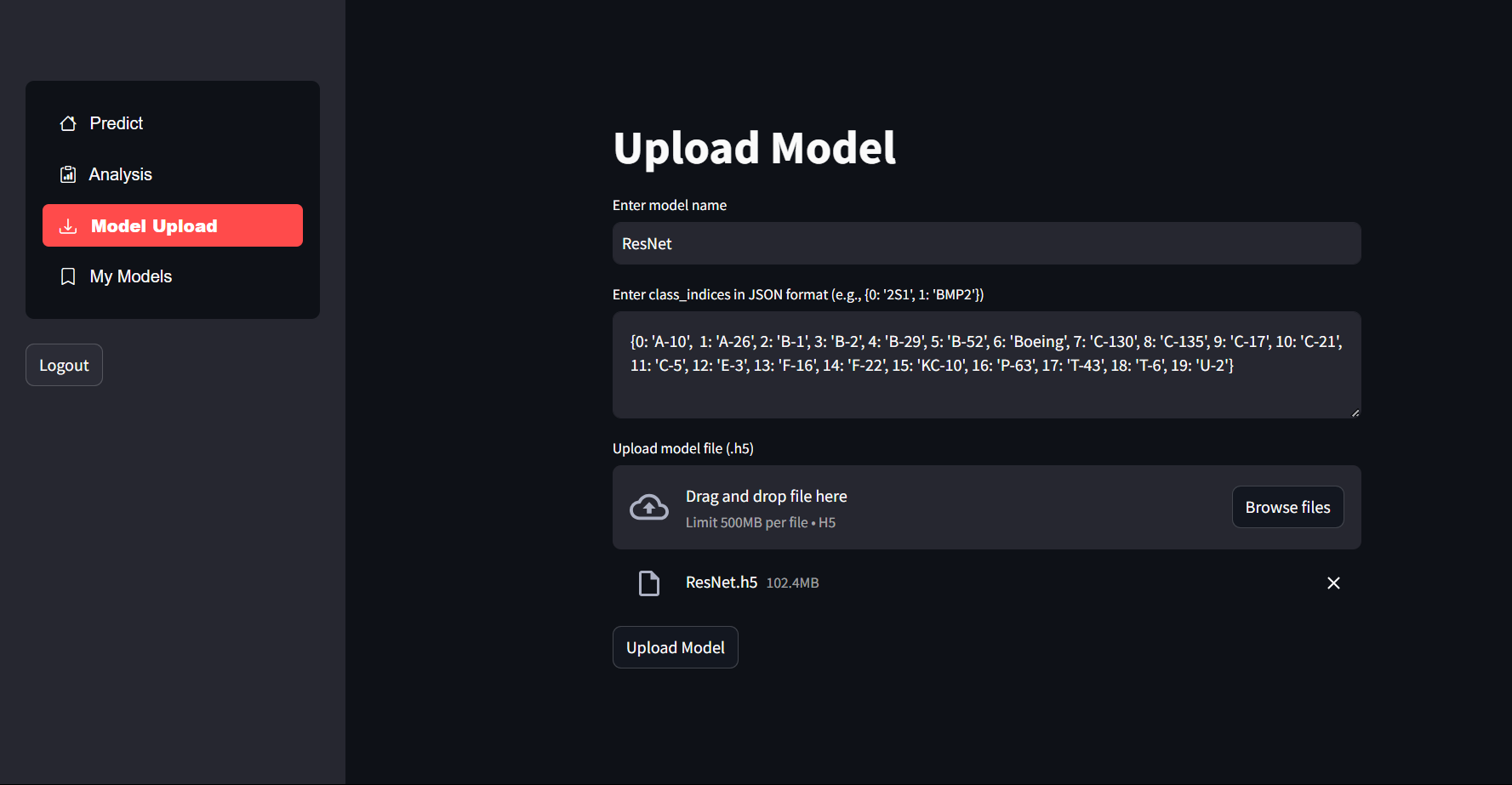


Рисунок 6.3 – Додавання нової моделі для прогнозів

Тепер, коли ми успішно завантажили нашу модель, ми можемо побачити її на сторінці управління моделями. Тут ми можемо видалити, або ж побачити її архітектуру (рис. 6.4 та рис. 6.5).

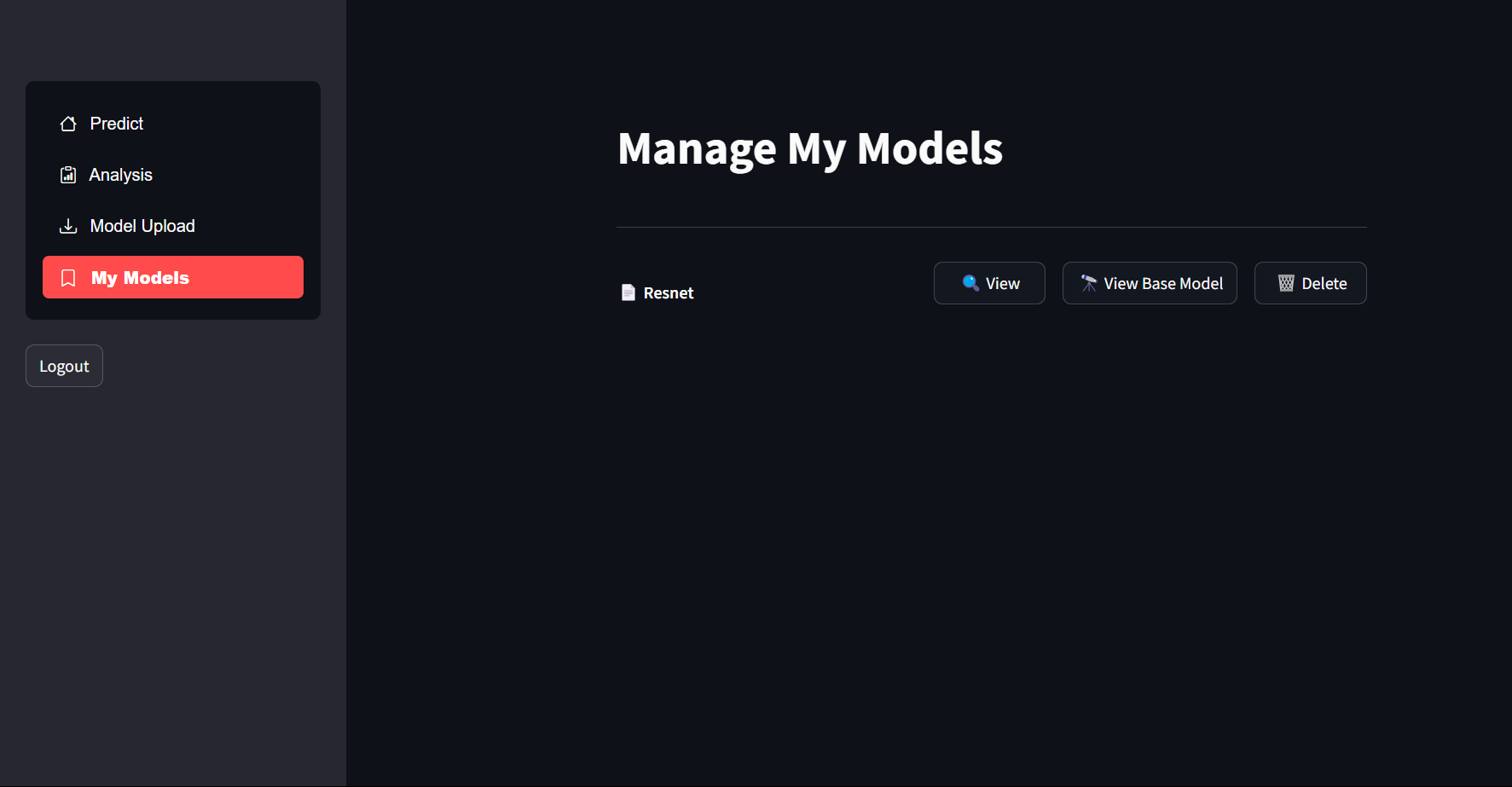


Рисунок 6.4 – Сторінка перегляду завантажених моделей

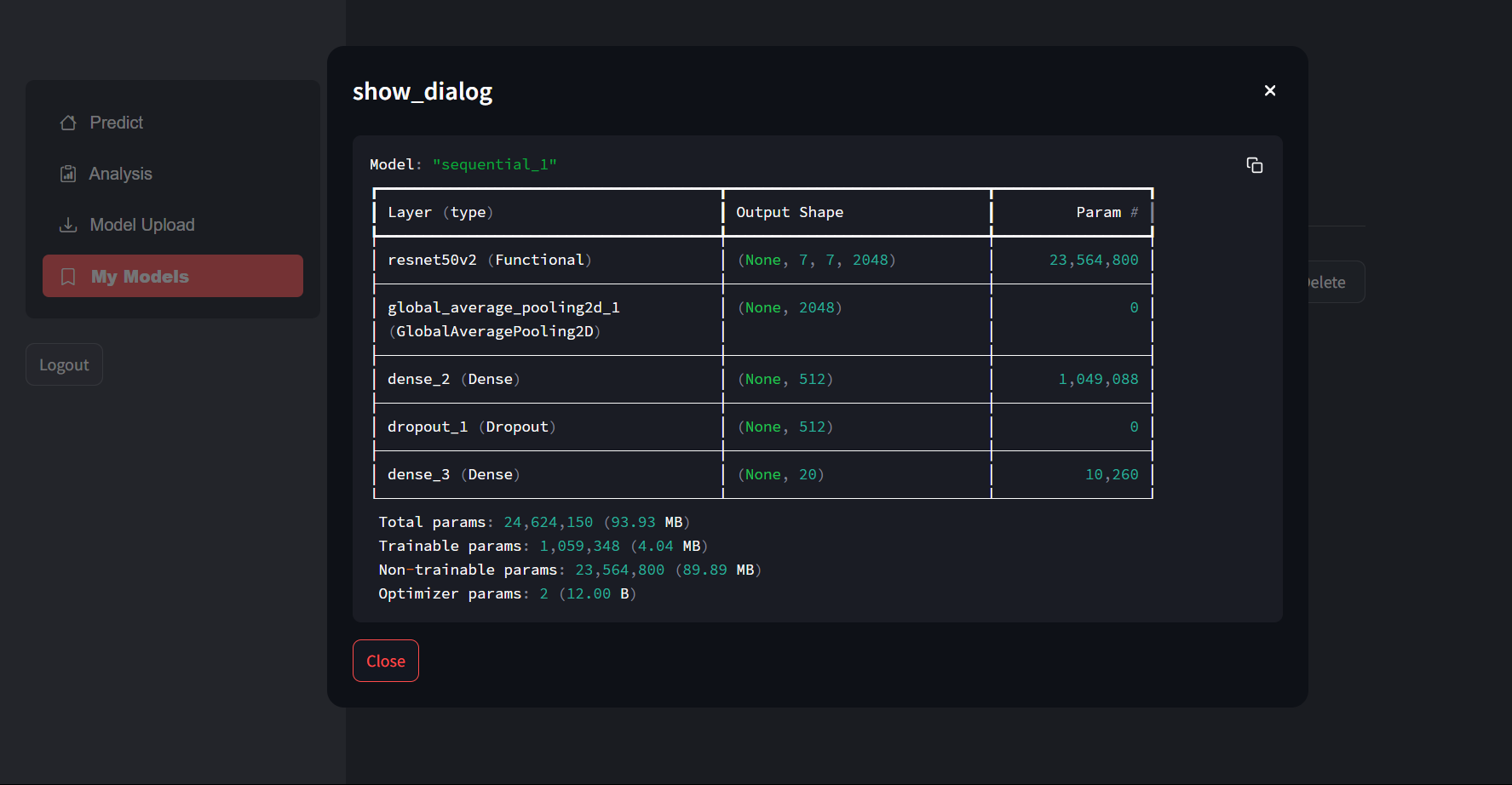


Рисунок 6.5 – Перегляд архітектури завантаженої моделі

Після того, як модель успішно завантажена, ми можемо використовувати її в наших обчисленнях. Перейдемо на основну сторінку, завантажимо зображення і поглянемо на результати на рисунку 6.6.

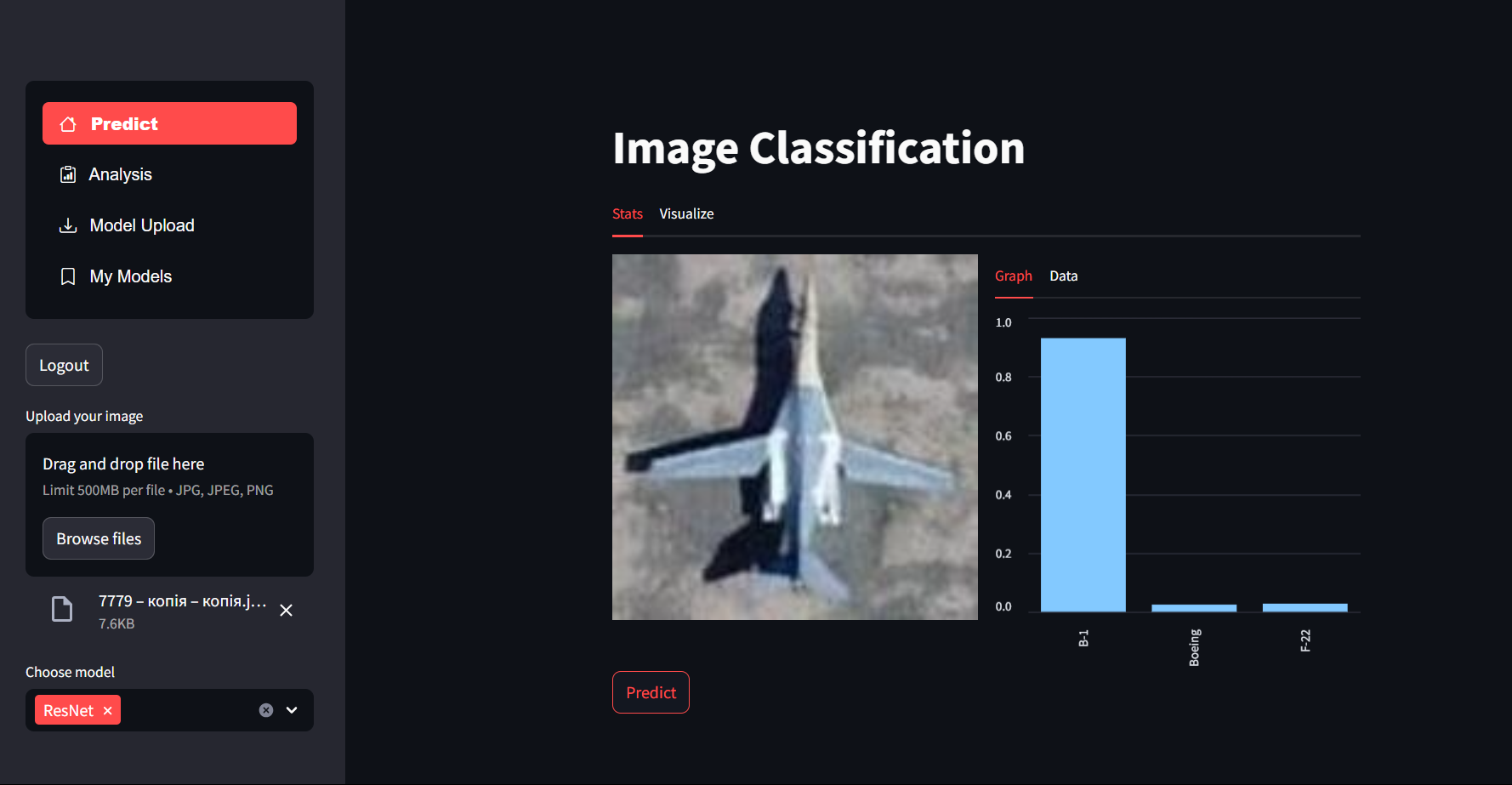


Рисунок 6.6 – Результати класифікації завантаженої моделі

Як видно з отриманих результатів, модель класифікувала даний літак як «B-1» з імовірністю, що перевищує 90%. Це свідчить про високу точність роботи моделі в даному випадку, що є важливим показником її ефективності. Крім того, ми маємо можливість згенерувати теплову карту, яка дозволяє візуалізувати найважливіші області зображення, що найбільше вплинули на прогноз моделі. Ця візуалізація є надзвичайно корисною, оскільки вона допомагає зрозуміти, які саме елементи зображення модель вважає критично важливими для прийняття рішення.

Тепер перейдемо на вкладку візуалізації, де ми можемо детальніше розглянути цю теплову карту (рис. 6.7). Вона надасть нам можливість проаналізувати, які частини зображення були найбільш значущими для моделі, а також виявити потенційні області для подальшого вдосконалення моделі. Це дозволяє не лише оцінити якість класифікації, але й отримати цінну інформацію для подальшого навчання та оптимізації моделі.

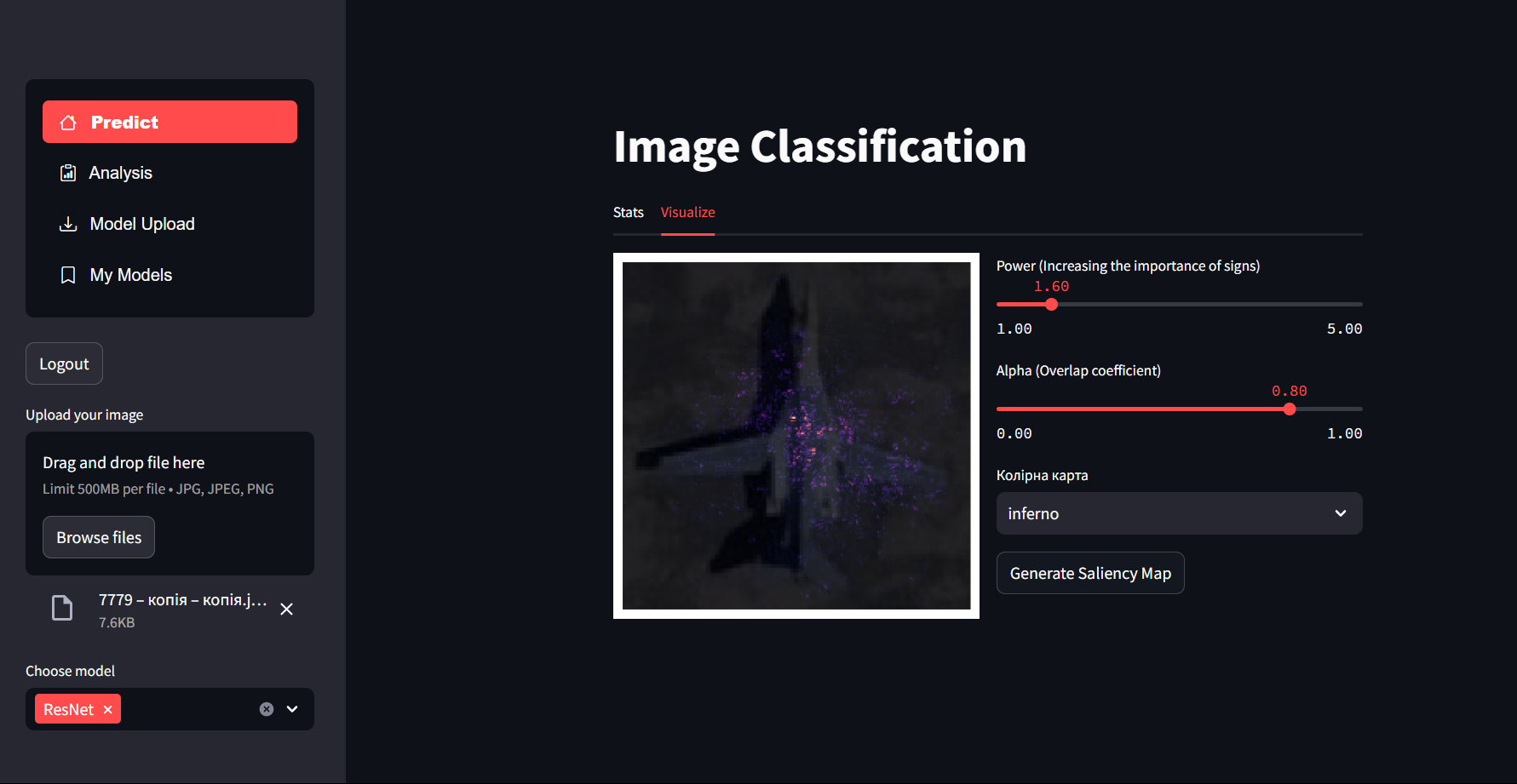


Рисунок 6.7 – Візуалізація найважливіших ознак

З рисунку видно, що найбільше на класифікацію вплинула центральна частина літака. Це вказує на те, що у цієї моделі може бути якась специфічна деталь, яка розташована саме в центральній частині корпусу літака. Ця деталь, ймовірно, є ключовою для розпізнавання типу літака, оскільки модель звертає на неї особливу увагу під час аналізу зображення. Така інформація може бути надзвичайно корисною для подальшого вдосконалення моделі, оскільки вона вказує на важливі ознаки, які слід враховувати при навчанні.

За описаним вище принципом, ми можемо додати інші види нейронних мереж, які також можуть бути використані для класифікації зображень. Це дозволить нам порівняти їхню ефективність та виявити, які з них найкраще підходять для конкретних завдань. На рисунку 6.8 представлені різні архітектури нейронних мереж, які можуть бути інтегровані в нашу систему, що відкриває нові можливості для покращення точності класифікації та розширення функціональності моделі.

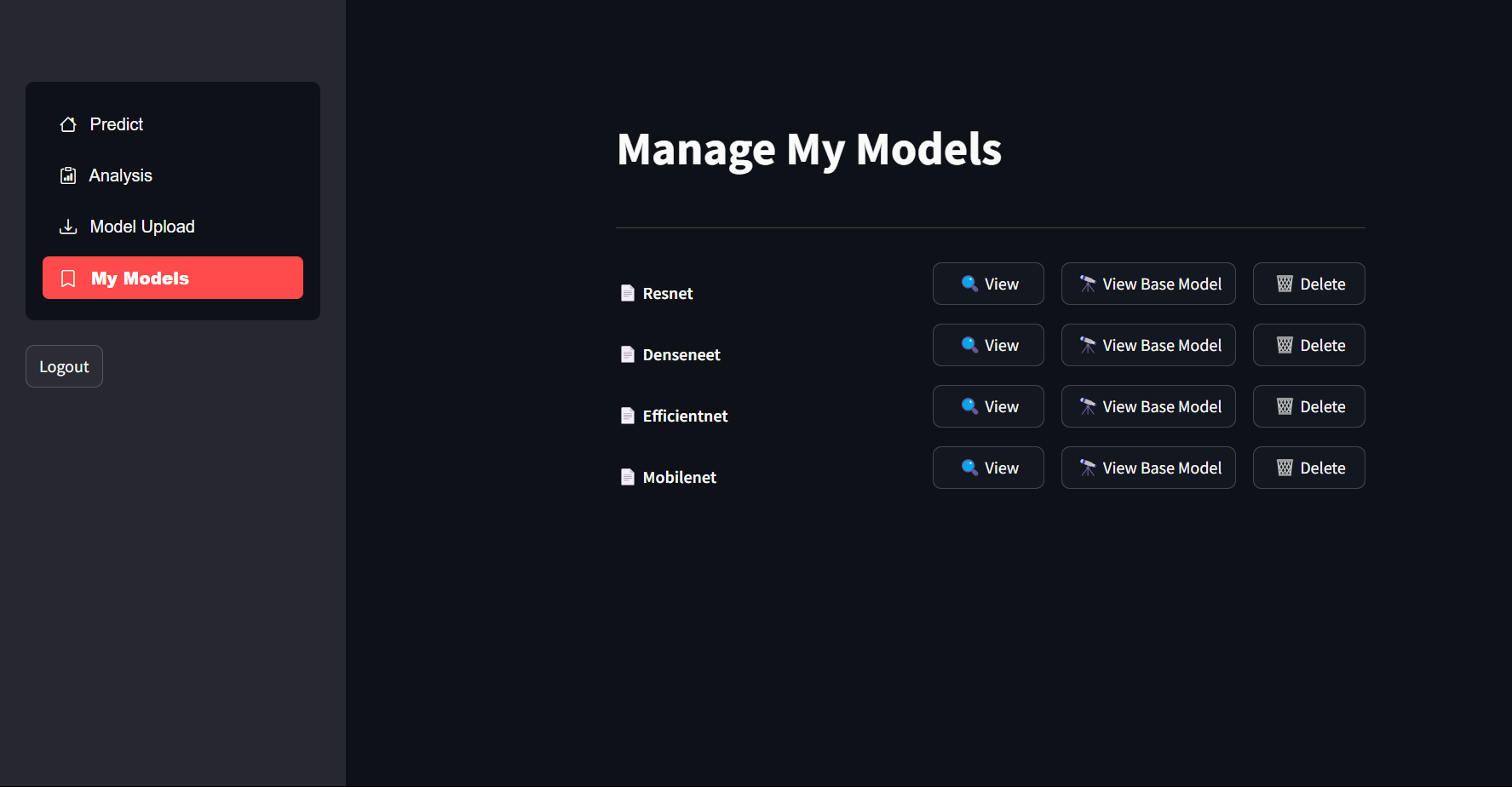


Рисунок 6.8 Завантаження додаткових моделей

Тепер ми можемо вибрати декілька моделей для порівняння прогнозів (рис. 6.9).

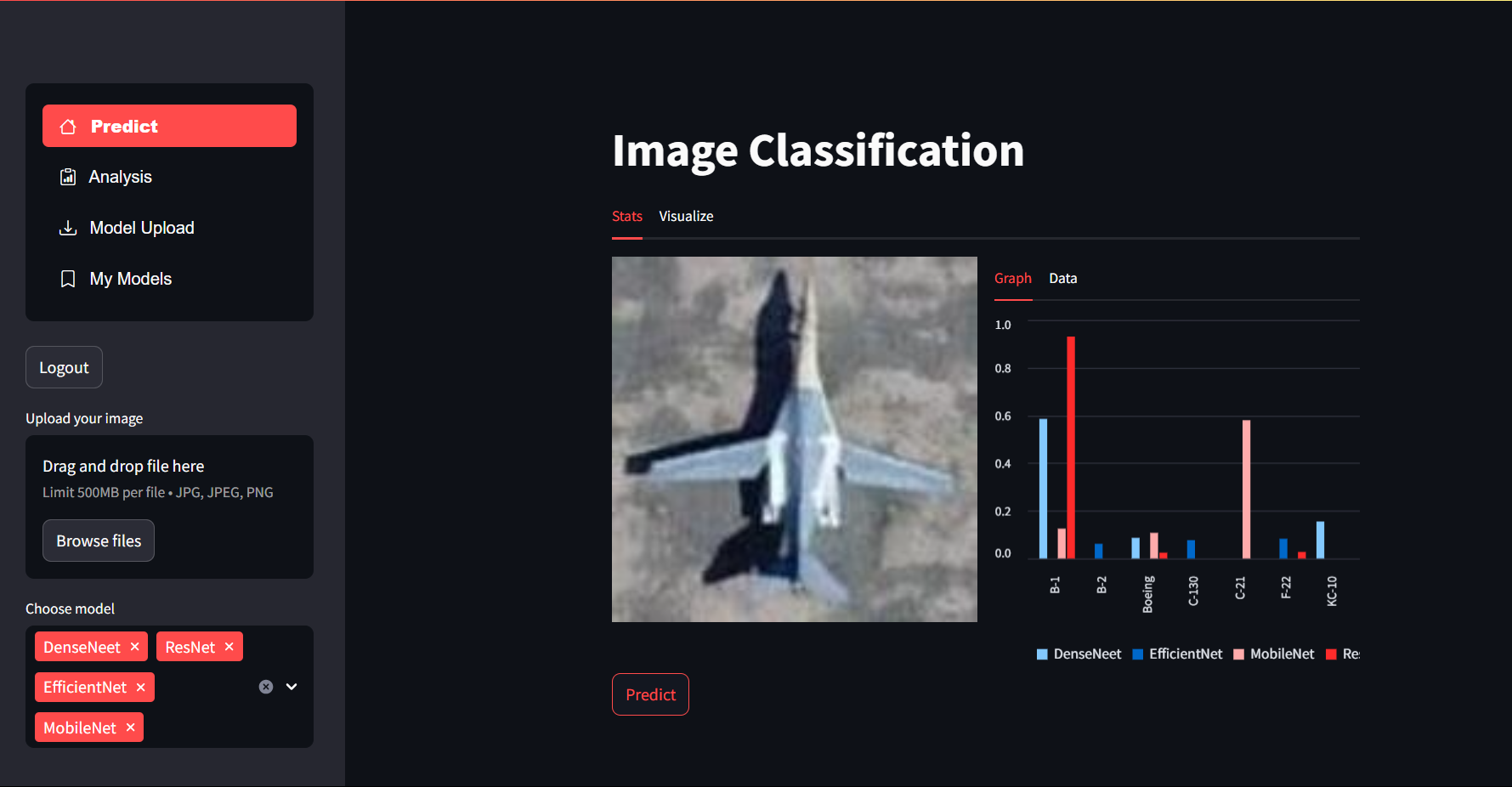


Рисунок 6.9 – Результати класифікації різних моделей

З рисунку видно, що більшість моделей визначили тип даного літака як «B-1»

На другій сторінці аналізу користувач має доступ до повного датасету значень класифікації для кожної моделі, що дозволяє детально оцінити точність і особливості роботи кожного алгоритму (рис. 6.10). З цього рисунка видно, як різні моделі класифікують зображення, що дає змогу порівняти їх продуктивність та ідентифікувати можливі розбіжності в оцінках. Такий підхід сприяє виявленню найкращої моделі для подальшого використання у специфічних задачах класифікації.

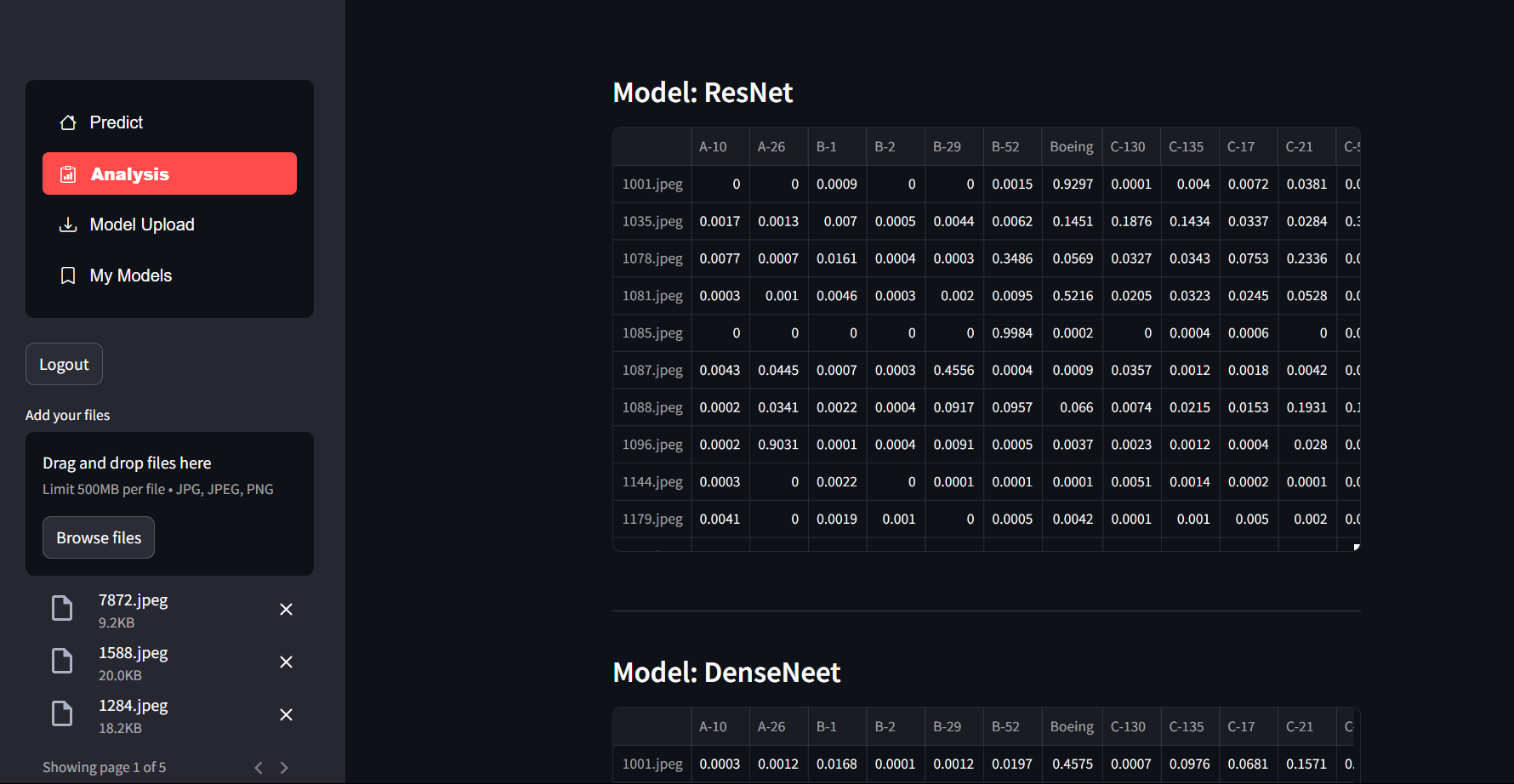


Рисунок 6.10 – Значення класифікацій моделей для кожного класу всіх зображень

Додатково представлена візуалізація важливих частин зображення для кожного файлу (рис. 6.11), що є ключовою особливістю системи. Ця візуалізація допомагає зрозуміти, які саме ознаки були найзначущими для прийняття рішення конкретною моделлю. Це дозволяє не лише оцінити ефективність роботи алгоритму, але й інтерпретувати його рішення, що є важливим аспектом для задач, де потрібна висока довіра до результатів класифікації.

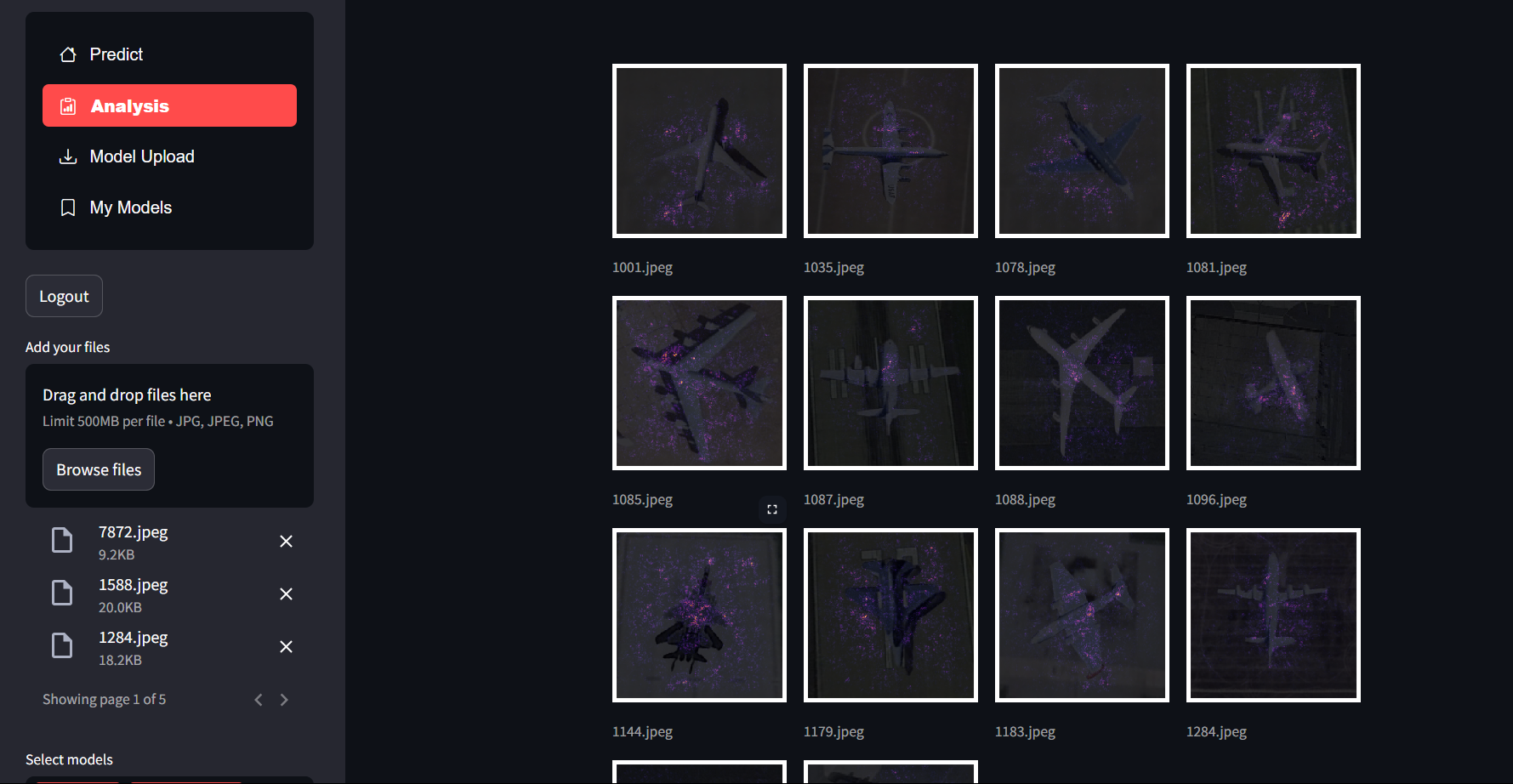


Рисунок 6.11 – Виявлення найважливіших ознак для кожного зображення

Отже, розроблена інформаційна система підтримки прийняття рішень успішно виконала задачі класифікації зображень на тестових прикладах, продемонструвавши свою функціональність, точність і зручність у використанні. Система дозволяє працювати з попередньо навченими моделями, виконувати прогнозування, візуалізувати результати у вигляді теплових карт та порівнювати ефективність різних моделей. Такий підхід забезпечує широкий спектр можливостей для користувача, зокрема аналіз результатів класифікації, інтерактивність та адаптивність до різних потреб. Представлені приклади наочно демонструють потенціал системи для використання у реальних завданнях класифікації зображень.

# Висновок

У ході виконання курсової роботи було створено інформаційну систему підтримки прийняття рішень для класифікації зображень, що забезпечує користувачам зручність, ефективність та адаптивність. Система демонструє високий рівень продуктивності, але, як і будь-яке програмне забезпечення, має свої переваги та недоліки.

Переваги розробленої системи:

* інтуїтивно зрозумілий інтерфейс, реалізований за допомогою Streamlit, дозволяє легко працювати з системою навіть користувачам із базовими технічними знаннями;
* підтримка завантаження користувацьких моделей нейронних мереж забезпечує гнучкість у застосуванні системи до різних задач;
* вбудовані інструменти візуалізації, такі як теплові карти, дозволяють аналізувати вплив окремих областей зображення на результати класифікації;
* система легко масштабується, що дає можливість обробляти як окремі зображення, так і пакети даних.

Недоліки системи:

* відсутність функцій автоматичного навчання моделей безпосередньо у системі. Користувач повинен заздалегідь підготувати та навчити модель у зовнішньому середовищі;
* продуктивність системи залежить від обчислювальних ресурсів, доступних на стороні користувача, що може обмежувати її застосування на пристроях із низькою потужністю;

Особливістю роботи системи є те, що вона орієнтована на дослідників, інженерів та фахівців, які працюють із класифікацією зображень у різних галузях. Вона забезпечує простий процес інтеграції користувацьких моделей, дозволяючи використовувати вже навчені нейронні мережі без необхідності написання складного коду.

Порівняно з іншими системами подібного призначення, розроблена ІСППР виділяється простотою у використанні. На відміну від комплексних платформ, таких як TensorFlow Serving або MLFlow, наша система не потребує складної інфраструктури для розгортання. Водночас, продуктивність обчислень відповідає сучасним стандартам, хоча функції автоматичного масштабування та оптимізації поки що відсутні.

На тестових прикладах система продемонструвала високу точність класифікації та зручність у використанні. Зокрема, прогнозування моделей, теплові карти та порівняльний аналіз забезпечили користувачеві глибокий інструментарій для аналізу даних. Система успішно впоралася з завданнями порівняння різних моделей і виявлення ключових ознак зображень, що впливають на результати класифікації.

Для покращення системи у майбутньому планується:

* додати можливість навчання моделей безпосередньо у системі, що забезпечить більшу автономність;
* оптимізувати роботу на пристроях із низькими обчислювальними ресурсами.
* інтегрувати можливість розгортання системи в хмарних середовищах для підвищення продуктивності та доступності;
* додати автоматизовані методи вибору найкращої моделі для класифікації на основі аналізу результатів.

Таким чином, розроблена система вже є ефективним інструментом для класифікації зображень, а подальший розвиток дозволить значно розширити її функціональність і сферу використання.

# Перелік посилань

1. Modular Software Architecture: A Review [Електронний ресурс] // Pretius. – Режим доступу: <https://pretius.com/blog/modular-software-architecture/>
2. Atchison, L. (2020). Architecting for Scale: High Availability for Your Growing Applications (2nd ed.). O'Reilly Media. ISBN 9781492057178
3. SIFT (Scale-Invariant Feature Transform) [Електронний ресурс] // Roboflow Blog. – Режим доступу: <https://blog.roboflow.com/sift/>
4. Histogram of Oriented Gradients [Електронний ресурс] // Built In. – Режим доступу: <https://builtin.com/articles/histogram-of-oriented-gradients>
5. Classification of Images Using Support Vector Machines [Електронний ресурс] / N. Cristianini, J. Shawe-Taylor, A. Elisseeff, J. Kandola // arXiv. – 2007. – Режим доступу: <https://arxiv.org/pdf/0709.3967>
6. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review [Електронний ресурс] // ResearchGate. – Режим доступу: <https://www.researchgate.net/publication/317496930_Deep_Convolutional_Neural_Networks_for_Image_Classification_A_Comprehensive_Review>
7. Deep Residual Learning for Image Recognition [Електронний ресурс] / K. He, X. Zhang, S. Ren, J. Sun // arXiv. – 2015. – Режим доступу: <https://arxiv.org/abs/1512.03385v1>
8. EfficientNet [Електронний ресурс] // Papers with Code. – Режим доступу: <https://paperswithcode.com/method/efficientnet>
9. DenseNet [Електронний ресурс] // Papers with Code. – Режим доступу: <https://paperswithcode.com/method/densenet>
10. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Електронний ресурс] / A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam // arXiv. – 2017. – Режим доступу: <https://arxiv.org/abs/1704.04861>

# Додатки

ДОДАТОК А

**Код файлу *main.py***

import bcrypt

import streamlit as st

from streamlit\_option\_menu import option\_menu

from pages.predict import show\_predict\_page

from pages.analysis import show\_models\_page

from pages.upload import show\_upload\_page

from pages.my\_models import show\_my\_models

from database import add\_user, get\_user

from database import create\_table, create\_models\_table

# Create tables

create\_table()

create\_models\_table()

# Initialize session state for persistent login

if 'logged\_in' not in st.session\_state:

st.session\_state['logged\_in'] = False

if 'username' not in st.session\_state:

st.session\_state['username'] = None

@st.cache\_resource

def get\_auth\_manager():

return {

"current\_user": None,

"auth\_status": False

}

auth\_manager = get\_auth\_manager()

def register\_user():

st.title("Registration")

with st.form("register\_form"):

username = st.text\_input("Enter username")

password = st.text\_input("Enter password", type="password")

confirm\_password = st.text\_input("Confirm password", type="password")

submit\_button = st.form\_submit\_button("Register")

if submit\_button:

if username and password and confirm\_password:

if password != confirm\_password:

st.error("Passwords do not match")

else:

user = get\_user(username)

if user:

st.error("User with this username already exists")

else:

add\_user(username, password)

st.success("Registration successful. You can now log in.")

else:

st.error("Please enter a username and password")

def login\_user():

st.title("Login")

with st.form("login\_form"):

username = st.text\_input("Enter username")

password = st.text\_input("Enter password", type="password")

submit\_button = st.form\_submit\_button("Login")

if submit\_button:

if username and password:

user = get\_user(username)

if user and bcrypt.checkpw(password.encode('utf-8'), user[2]):

st.session\_state['logged\_in'] = True

st.session\_state['username'] = username

auth\_manager["current\_user"] = username

auth\_manager["auth\_status"] = True

st.rerun()

else:

st.error("Invalid username or password")

else:

st.error("Please enter a username and password")

# Check if user is already authenticated

if auth\_manager["auth\_status"]:

st.session\_state['logged\_in'] = True

st.session\_state['username'] = auth\_manager["current\_user"]

# Main application logic

if st.session\_state['logged\_in']:

with st.sidebar:

selected = option\_menu(

menu\_title=None,

options=["Predict", "Analysis", "Model Upload", "My Models"],

icons=["house", "clipboard-data", "download", "bookmark"]

)

if st.button("Logout"):

st.session\_state['logged\_in'] = False

st.session\_state['username'] = None

auth\_manager["current\_user"] = None

auth\_manager["auth\_status"] = False

st.rerun()

if selected == "Predict":

show\_predict\_page()

elif selected == "Analysis":

show\_models\_page()

elif selected == "Model Upload":

show\_upload\_page()

elif selected == "My Models":

show\_my\_models()

else:

auth\_option = st.sidebar.radio("Choose an option", ["Login", "Register"])

if auth\_option == "Login":

login\_user()

elif auth\_option == "Register":

register\_user()

ДОДАТОК Б

**Код файлу *database.py***

import psycopg2.pool

import bcrypt

import json

import os

import streamlit as st

connection\_pool = psycopg2.pool.SimpleConnectionPool(

minconn=1,

maxconn=10,

host="localhost",

database="test",

user="postgres",

password="root"

)

# Connect to database

def get\_connection():

return connection\_pool.getconn()

# Disconect

def release\_connection(conn):

connection\_pool.putconn(conn)

def create\_table():

conn = get\_connection()

try:

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS users (

id SERIAL PRIMARY KEY,

username TEXT NOT NULL UNIQUE,

password BYTEA NOT NULL)''')

conn.commit()

except (Exception, psycopg2.Error) as error:

print(f"Помилка при створенні таблиці users: {error}")

finally:

if cursor:

cursor.close()

release\_connection(conn)

def create\_models\_table():

conn = get\_connection()

try:

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS models (

id SERIAL PRIMARY KEY,

user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,

model\_name TEXT NOT NULL,

model\_path TEXT NOT NULL,

class\_indices JSON NOT NULL)''') # Removed BYTEA

conn.commit()

except (Exception, psycopg2.Error) as error:

print(f"Помилка при створенні таблиці models: {error}")

finally:

if cursor:

cursor.close()

release\_connection(conn)

def add\_user(username, password):

try:

conn = get\_connection()

cursor = conn.cursor()

hashed\_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

cursor.execute('INSERT INTO users (username, password) VALUES (%s, %s)',

(username, hashed\_password))

conn.commit()

except (Exception, psycopg2.Error) as error:

print(f"Помилка при додаванні користувача: {error}")

raise

finally:

if cursor:

cursor.close()

release\_connection(conn)

def get\_user(username):

try:

conn = get\_connection()

cursor = conn.cursor()

cursor.execute('SELECT \* FROM users WHERE username = %s', (username,))

user = cursor.fetchone()

if user:

# Конвертуємо memoryview в bytes для паролю

return (user[0], user[1], bytes(user[2]))

return user

except (Exception, psycopg2.Error) as error:

print(f"Помилка при отриманні користувача: {error}")

return None

finally:

if cursor:

cursor.close()

release\_connection(conn)

def add\_model(user\_id, model\_name, class\_indices, model\_path):

try:

conn = get\_connection()

cursor = conn.cursor()

full\_model\_name = f"{model\_name}\_{st.session\_state['username']}.h5" # Naming convention

full\_model\_path = os.path.join(model\_path, full\_model\_name) # Complete file path

cursor.execute('INSERT INTO models (user\_id, model\_name, model\_path, class\_indices) VALUES (%s, %s, %s, %s)',

(user\_id, model\_name, full\_model\_path, json.dumps(class\_indices)))

conn.commit()

return full\_model\_path # Return the path for saving the file

except (Exception, psycopg2.Error) as error:

print(f"Помилка при додаванні моделі: {error}")

raise

finally:

if cursor:

cursor.close()

release\_connection(conn)

def remove\_model\_from\_db(model\_name):

user = get\_user(st.session\_state['username'])

user\_id = user[0]

try:

# Отримання з'єднання з базою даних

conn = get\_connection()

cursor = conn.cursor()

delete\_query = """

DELETE FROM models

WHERE user\_id = %s AND model\_name = %s

"""

cursor.execute(delete\_query, (user\_id, model\_name))

conn.commit()

return True

except (Exception, psycopg2.Error) as error:

print(f"Помилка при видаленні моделі '{model\_name}' для користувача з ID {user\_id}: {error}")

if conn:

conn.rollback() # Відкочування транзакції у випадку помилки

return False

finally:

if cursor:

cursor.close()

if conn:

release\_connection(conn)

def get\_models(user\_id):

try:

conn = get\_connection()

cursor = conn.cursor()

cursor.execute('SELECT model\_name, class\_indices, model\_path FROM models WHERE user\_id = %s', (user\_id,))

models = cursor.fetchall()

return models

except (Exception, psycopg2.Error) as error:

print(f"Помилка при отриманні моделей: {error}")

return []

finally:

if cursor:

cursor.close()

release\_connection(conn)

# Creating a table at startup

create\_table()

create\_models\_table()

ДОДАТОК В

**Код файлу *predict.py***

import streamlit as st

from PIL import Image

import numpy as np

import pandas as pd

from database import get\_user, get\_models

from utils import \*

def show\_predict\_page():

st.write("# Image Classification")

# Initialize session state variables if not already set

if 'uploaded\_file' not in st.session\_state:

st.session\_state.uploaded\_file = None

if 'all\_predictions' not in st.session\_state:

st.session\_state.all\_predictions = []

# Fetch models only once - use the working version from the second page

user = get\_user(st.session\_state['username'])

if not user:

st.error("No user found")

return

models\_df = pd.DataFrame(get\_models(user[0]), columns=['model\_name', 'class\_indices', 'model\_path'])

models = {name: load\_model(path) for name, path in

models\_df.set\_index('model\_name')['model\_path'].items()}

# Sidebar file uploader and model selection

uploaded\_file = st.sidebar.file\_uploader("Upload your image", type=["jpg", "jpeg", "png"], key='shared\_uploader')

# Unified model multiselect for both tabs

selected\_models = st.sidebar.multiselect(

"Choose model",

list(models.keys())

)

# Create tabs

col\_stats, col\_visualize = st.tabs(["Stats", "Visualize"])

# Update uploaded file in session state

if uploaded\_file is not None:

st.session\_state.uploaded\_file = uploaded\_file

with col\_stats:

if st.session\_state.uploaded\_file is not None:

image = Image.open(st.session\_state.uploaded\_file)

col1, col2 = st.columns([1, 1])

with col1:

st.image(image, use\_container\_width=True)

st.text("")

st.text("")

btn\_predict = st.button('Predict')

with col2:

if btn\_predict:

if not selected\_models:

st.toast('Please select at least one model to predict.', icon="🚨")

else:

st.session\_state.all\_predictions.clear() # Clear previous predictions

for model\_name in selected\_models:

# Get class indices for current model

class\_indices = models\_df.loc[

models\_df['model\_name'] == model\_name,

'class\_indices'

].values[0]

if isinstance(class\_indices, str):

class\_indices = eval(class\_indices)

predictions = predict(image, models[model\_name])

top\_3\_indices = np.argsort(predictions[0])[-3:][::-1]

top\_3\_predictions = [(class\_indices[str(index)], predictions[0][index]) for index in top\_3\_indices]

st.session\_state.all\_predictions.append((model\_name, top\_3\_predictions))

# Visualization of predictions

Graph, Data = st.tabs(["Graph", "Data"])

with Graph:

if st.session\_state.all\_predictions:

# Prepare chart data

chart\_data = pd.DataFrame(columns=['Model', 'Class', 'Probability'])

for model\_name, top\_3\_predictions in st.session\_state.all\_predictions:

for class\_name, probability in top\_3\_predictions:

new\_row = pd.DataFrame({

'Model': [model\_name],

'Class': [class\_name],

'Probability': [probability]

})

chart\_data = pd.concat([chart\_data, new\_row], ignore\_index=True)

# Plot bar chart

if len(selected\_models) == 1:

chart\_data = chart\_data.set\_index('Class')

st.bar\_chart(chart\_data[['Probability']])

else:

chart\_data = chart\_data.pivot(index='Class', columns='Model', values='Probability')

st.bar\_chart(chart\_data, stack=False)

with Data:

if 'chart\_data' in locals():

st.write(chart\_data)

else:

st.info('Upload your image for prediction')

with col\_visualize:

if st.session\_state.uploaded\_file is not None and selected\_models:

# Use first selected model for saliency map

selected\_model = selected\_models[0]

# Ensure image is preprocessed for saliency map

image, original\_size = load\_and\_preprocess\_image(

st.session\_state.uploaded\_file,

model=models[selected\_model]

)

col1, col2 = st.columns(2)

with col2:

power = st.slider('Power (Increasing the importance of signs)', 1.0, 5.0, 2.0, 0.1)

alpha = st.slider('Alpha (Overlap coefficient)', 0.0, 1.0, 0.7, 0.1)

colormap\_name = st.selectbox('Колірна карта', [

'coolwarm', 'viridis', 'plasma', 'inferno', 'cividis'

])

if st.button('Generate Saliency Map'):

# Ensure image is preprocessed for saliency map

image, original\_size = load\_and\_preprocess\_image(

st.session\_state.uploaded\_file,

model=models[selected\_model]

)

saliency\_map, predicted\_class = generate\_saliency\_map(

models[selected\_model],

image

)

st.session\_state['saliency\_map'] = saliency\_map

st.session\_state['predicted\_class'] = predicted\_class

with col1:

# Display saliency map

if 'saliency\_map' in st.session\_state and st.session\_state['saliency\_map'] is not None:

fig = visualize\_saliency\_on\_image(

image,

st.session\_state['saliency\_map'],

alpha=alpha,

power=power,

colormap\_name=colormap\_name

)

st.pyplot(fig)

plt.close(fig)

def get\_models\_dict\_and\_df():

user = get\_user(st.session\_state['username'])

if not user:

st.error("No user found")

return {}, pd.DataFrame()

models\_df = pd.DataFrame(get\_models(user[0]), columns=['model\_name', 'class\_indices', 'model\_path'])

models = {}

for \_, row in models\_df.iterrows():

models[row['model\_name']] = load\_model(row['model\_path'])

return models, models\_df

ДОДАТОК Г

**Код файлу *analysis.py***

import pandas as pd

import streamlit as st

from PIL import Image

from utils import \*

from database import get\_user, get\_models

def show\_models\_page():

files = st.sidebar.file\_uploader("Add your files", accept\_multiple\_files=True, type=["jpg", "jpeg", "png"])

models, models\_df = get\_models\_dict\_and\_df()

selected\_models = st.sidebar.multiselect("Select models", list(models.keys()))

# Create tabs

tab1, tab2 = st.tabs(["Datasets", "Saliency Maps"])

if files and selected\_models:

with tab1:

# Create predictions for each model

for model\_name in selected\_models:

st.subheader(f"Model: {model\_name}")

# Get class indices

class\_indices = models\_df.loc[models\_df['model\_name'] == model\_name, 'class\_indices'].values[0]

if isinstance(class\_indices, str):

class\_indices = eval(class\_indices)

class\_names = [class\_indices[str(i)] for i in range(len(class\_indices))]

# Create DataFrame

results = []

file\_names = []

for file in files:

image = Image.open(file)

prediction = predict(image, models[model\_name])

results.append(prediction)

file\_names.append(file.name)

# Create and display DataFrame

probabilities\_df = pd.DataFrame(columns=class\_names)

for file, result in zip(file\_names, results):

row = pd.Series(result.flatten(), index=class\_names, name=file)

probabilities\_df = pd.concat([probabilities\_df, row.to\_frame().T], ignore\_index=False)

st.dataframe(probabilities\_df)

st.markdown("---")

with tab2:

# Saliency controls in main area

power = st.slider('Power (Intensity)', 1.0, 5.0, 2.0, 0.1)

alpha = st.slider('Alpha (Overlay)', 0.0, 1.0, 0.4, 0.1)

colormap = st.selectbox('Colormap', ['jet', 'viridis', 'plasma', 'inferno', 'cividis'])

col1, col2 = st.columns(2)

with col2:

# Button to generate saliency maps

generate\_button = st.button("Generate Saliency Maps")

# Store parameters to persist across button presses

st.session\_state['saliency\_params'] = {

'power': power,

'alpha': alpha,

'colormap': colormap

}

with col1:

# Only process and show saliency maps when button is pressed

if generate\_button:

# Validate that files and models are selected

if not files or not selected\_models:

st.warning("Please upload files and select models")

else:

# Use the first selected model

model = models[selected\_models[0]]

processed\_images = []

for file in files:

image = Image.open(file)

processed\_image, \_ = load\_and\_preprocess\_image(file, model=model)

saliency\_map, predicted\_class = generate\_saliency\_map(model, processed\_image)

fig = visualize\_saliency\_on\_image(

processed\_image,

saliency\_map,

alpha=st.session\_state['saliency\_params']['alpha'],

power=st.session\_state['saliency\_params']['power'],

colormap\_name=st.session\_state['saliency\_params']['colormap']

)

processed\_images.append((fig, file.name))

# Store processed images in session state

st.session\_state['processed\_images'] = processed\_images

# Display grid of processed images only when they exist in session state

if 'processed\_images' in st.session\_state:

for i in range(0, len(st.session\_state['processed\_images']), 4):

cols = st.columns(4)

for j, col in enumerate(cols):

if i + j < len(st.session\_state['processed\_images']):

with col:

fig, filename = st.session\_state['processed\_images'][i + j]

st.pyplot(fig, use\_container\_width=True)

plt.close(fig)

st.caption(filename)

else:

st.warning("Please upload files and select models")

def get\_models\_dict\_and\_df():

user = get\_user(st.session\_state['username'])

if not user:

st.error("No user found")

return {}, pd.DataFrame()

models\_df = pd.DataFrame(get\_models(user[0]), columns=['model\_name', 'class\_indices', 'model\_path'])

models = {name: load\_model(path) for name, path in

models\_df.set\_index('model\_name')['model\_path'].items()}

return models, models\_df

ДОДАТОК Д

**Код файлу *my\_models.py***

import streamlit as st

import pandas as pd

import io

import os

from database import get\_user, get\_models, remove\_model\_from\_db

from utils import \*

def show\_my\_models():

st.title("Manage My Models")

st.divider()

user = get\_user(st.session\_state['username'])

if user:

models\_df = pd.DataFrame(get\_models(user[0]), columns=['model\_name', 'class\_indices', 'model\_path'])

for \_, row in models\_df.iterrows():

model\_name, model\_path = row['model\_name'], row['model\_path']

if not os.path.exists(model\_path):

st.error(f"Файл моделі '{model\_path}' не знайдено.")

continue

with st.container():

col\_name, col\_button1, col\_button2, col\_button3 = st.columns([5, 2, 3, 2])

with col\_name:

st.write("")

st.write(f"📄 \*\*{model\_name.capitalize()}\*\*")

with col\_button1:

if st.button("🔍 View", key=model\_name,

use\_container\_width=True):

show\_model(model\_path)

with col\_button2:

if st.button("🔭 View Base Model", key=f"{model\_name}\_base",

use\_container\_width=True):

show\_model(model\_path, base\_model=True)

with col\_button3:

if st.button("🗑️ Delete", key=f"{model\_name}\_delete",

use\_container\_width=True):

@st.dialog(f"🚨 Ви впевнені, що хочете видалити модель '{model\_name.capitalize()}'?")

def delete\_model\_dialog():

st.write("")

st.html("<span class='big-dialog'></span>")

col1, col2, \_ = st.columns([1, 1, 3])

with col1:

if st.button("No", use\_container\_width=True):

st.rerun()

with col2:

if st.button("Yes", use\_container\_width=True):

delete\_model(model\_name, model\_path)

st.rerun()

delete\_model\_dialog()

def show\_model(model\_path, base\_model=False):

summary = io.StringIO()

model = load\_model(model\_path)

model = model if not base\_model else model.layers[0]

model.summary(print\_fn=lambda x: summary.write(x + '\n'))

summary\_str = summary.getvalue()

st.markdown(

"""

<style>

div[data-testid="stDialog"] div[role="dialog"]:has(.big-dialog) {

width: 60vw;

}

</style>

""",

unsafe\_allow\_html=True,

)

@st.dialog("show\_dialog")

def show\_dialog():

st.code(summary\_str, language='python')

st.html("<span class='big-dialog'></span>")

if st.button("Close"):

st.rerun()

show\_dialog()

def delete\_model(model\_name, model\_path):

try:

if os.path.exists(model\_path):

os.remove(model\_path)

remove\_model\_from\_db(model\_name)

st.rerun()

st.success(f"✅ Модель '{model\_name}' успішно видалена.")

else:

st.error(f"❌ Файл моделі '{model\_path}' не знайдено.")

except Exception as e:

st.error(f"❌ Сталася помилка при видаленні моделі '{model\_name}': {e}")

ДОДАТОК Е

**Код файлу *upload.py***

import os

import pandas as pd

import streamlit as st

from database import add\_model, get\_user, get\_models

def show\_upload\_page():

st.title("Upload Model")

# Check if user is logged in

if 'logged\_in' not in st.session\_state or not st.session\_state['logged\_in']:

st.error("Please log in to upload a model.")

return

# Model name input

model\_name = st.text\_input("Enter model name")

# Class indices input

class\_indices\_input = st.text\_area("Enter class\_indices in JSON format (e.g., {0: '2S1', 1: 'BMP2'})")

# Upload model file

uploaded\_model = st.file\_uploader("Upload model file (.h5)", type=["h5"])

if st.button("Upload Model"):

if not model\_name:

st.error("Please enter a model name.")

elif not class\_indices\_input:

st.error("Please enter class\_indices.")

elif not uploaded\_model:

st.error("Please upload a model file.")

else:

# Validate class\_indices format

try:

class\_indices = eval(class\_indices\_input)

if not isinstance(class\_indices, dict):

st.error("class\_indices must be a dictionary.")

return

if not all(isinstance(v, str) for v in class\_indices.values()):

st.error("All values in the dictionary must be strings.")

return

except (SyntaxError, NameError, TypeError):

st.error("Invalid Python dictionary format.")

return

# Get user ID

user = get\_user(st.session\_state['username'])

if not user:

st.error("User not found.")

return

user\_id = user[0]

models\_df = pd.DataFrame(get\_models(user[0]), columns=['model\_name', 'model\_path', 'class\_indices'])

if not models\_df.empty and model\_name in models\_df['model\_name'].values:

st.error(f"Model with name '{model\_name}' already exists. Please choose a different name.")

return

# Define directory for saving models

models\_dir = os.path.join("user\_models", f"user\_{st.session\_state['username']}")

os.makedirs(models\_dir, exist\_ok=True)

full\_model\_name = f"{model\_name}\_{st.session\_state['username']}.h5"

model\_path = os.path.join(models\_dir, full\_model\_name)

# Save uploaded model to filesystem

try:

with open(model\_path, "wb") as f:

f.write(uploaded\_model.getbuffer())

except Exception as e:

st.error(f"An error occurred while saving the model file: {e}")

return

# Add model to database

try:

add\_model(user\_id, model\_name, class\_indices, models\_dir)

st.success("Model successfully uploaded.")

except Exception as e:

st.error(f"An error occurred while adding the model to the database: {e}")

# Delete saved file if database insertion failed

if os.path.exists(model\_path):

os.remove(model\_path)

st.rerun()

ДОДАТОК Ж

**Код файлу *upload.py***

import os

import pandas as pd

import streamlit as st

from database import add\_model, get\_user, get\_models

def show\_upload\_page():

st.title("Upload Model")

# Check if user is logged in

if 'logged\_in' not in st.session\_state or not st.session\_state['logged\_in']:

st.error("Please log in to upload a model.")

return

# Model name input

model\_name = st.text\_input("Enter model name")

# Class indices input

class\_indices\_input = st.text\_area("Enter class\_indices in JSON format (e.g., {0: '2S1', 1: 'BMP2'})")

# Upload model file

uploaded\_model = st.file\_uploader("Upload model file (.h5)", type=["h5"])

if st.button("Upload Model"):

if not model\_name:

st.error("Please enter a model name.")

elif not class\_indices\_input:

st.error("Please enter class\_indices.")

elif not uploaded\_model:

st.error("Please upload a model file.")

else:

# Validate class\_indices format

try:

class\_indices = eval(class\_indices\_input)

if not isinstance(class\_indices, dict):

st.error("class\_indices must be a dictionary.")

return

if not all(isinstance(v, str) for v in class\_indices.values()):

st.error("All values in the dictionary must be strings.")

return

except (SyntaxError, NameError, TypeError):

st.error("Invalid Python dictionary format.")

return

# Get user ID

user = get\_user(st.session\_state['username'])

if not user:

st.error("User not found.")

return

user\_id = user[0]

models\_df = pd.DataFrame(get\_models(user[0]), columns=['model\_name', 'model\_path', 'class\_indices'])

if not models\_df.empty and model\_name in models\_df['model\_name'].values:

st.error(f"Model with name '{model\_name}' already exists. Please choose a different name.")

return

# Define directory for saving models

models\_dir = os.path.join("user\_models", f"user\_{st.session\_state['username']}")

os.makedirs(models\_dir, exist\_ok=True)

full\_model\_name = f"{model\_name}\_{st.session\_state['username']}.h5"

model\_path = os.path.join(models\_dir, full\_model\_name)

# Save uploaded model to filesystem

try:

with open(model\_path, "wb") as f:

f.write(uploaded\_model.getbuffer())

except Exception as e:

st.error(f"An error occurred while saving the model file: {e}")

return

# Add model to database

try:

add\_model(user\_id, model\_name, class\_indices, models\_dir)

st.success("Model successfully uploaded.")

except Exception as e:

st.error(f"An error occurred while adding the model to the database: {e}")

# Delete saved file if database insertion failed

if os.path.exists(model\_path):

os.remove(model\_path)

st.rerun()