VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Artificial Intelligence
IT159IU
PROJECT REPORT

Topic: Galaxy Image Classification System

By Group 1ConMeo– Member List
Nguyễn Đình Khánh Ngân - ITCSIU22236
Instructor: Nguyen Trung Ky

TABLE OF CONTENTS

ABSTRACT

The Galaxy Morphology Classification project introduces a deep-learning pipeline that automatically categorises optical images of galaxies into ten morphological classes defined by the **Galaxy10 DECals** dataset. Two Convolutional Neural Network (CNN) models—EfficientNet-B0 (hereafter *Model Old*) and EfficientNet-B2 (*Model New*)—were trained independently, each surpassing 80 % accuracy on training, validation, and test splits. Although neither model has yet reached the 90 % threshold, their complementary strengths motivate an ensemble strategy. The trained models are exposed through a Flask API and consumed by a React front-end that allows users to upload images and receive real-time predictions. This report details the dataset, system architecture, related work, training methodology, implementation specifics, empirical results, evaluation, and future improvements.

CHAPTER 1

INTRODUCTION

1.1 Project overview

Modern sky surveys capture millions of galaxy images daily. Accurate automatic classification of galaxy morphology ("spiral", "elliptical", "edge-on", ..) aids astronomers in studying structure formation, galaxy evolution, and large-scale cosmic structure. The Galaxy10-DECaLS dataset (Zenodo, 2023) contains 17,736 RGB images (256×256 pixels) labeled into ten morphology classes:

1. Disturbed Galaxy
2. Merging Galaxy
3. Round Smooth Galaxy
4. Intermediate Smooth Galaxy
5. Cigar-shaped Smooth Galaxy
6. Barred Spiral Galaxy
7. Unbarred Tight Spiral Galaxy
8. Unbarred Loose Spiral Galaxy
9. Edge-on Galaxy (no bulge)
10. Edge-on Galaxy (with bulge)

Galactic morphology is a key observable for understanding the formation and evolution of galaxies. Manual classification—historically done by experts or citizen-science initiatives like *Galaxy Zoo*—is slow and labour intensive. Leveraging recent advances in computer vision, our project delivers an end-to-end system that classifies raw telescope images into ten established categories (e.g., *Merging Galaxy*, *Barred Spiral Galaxy*) with a user-friendly web interface.

1.2 Problem Statement

Existing manual or semi-automated methods for morphological classification cannot scale to the extensive datasets produced by modern surveys (e.g., SDSS, DECals). Deep-learning models have shown promise, but single architectures often plateau near mid-80 % accuracy due to class imbalance and subtle visual distinctions. There is a need for a robust, modular system that combines complementary models to improve classification accuracy and delivers results through a scalable micro-service architecture

1.3 Objective & Scope

- **Objective 1:** Achieve ≥ 85 % accuracy (train/val/test) on the Galaxy10 DECals dataset using independent CNN models.
- **Objective 2:** Design an **ensemble mechanism** that reconciles model disagreement and aims toward ≥ 90 % aggregate accuracy.
- **Objective 3:** Deploy the solution as a RESTful service with an intuitive React interface.
  The scope is limited to single-image, single-label inference. Tasks such as red-shift estimation, multi-label tagging, or active-learning feedback are considered future extensions.

CHAPTER 2

A PART OFRELATIVE WORK

1. Traditional Methods
    - Feature Engineering: Early work extracted hand-crafted features (e.g., shape descriptors, texture indices) and applied classical classifiers (SVM, Random Forest). While interpretable, these methods struggled with complex morphological variation and required domain expertise to design features.
    - Principal Component Analysis (PCA) & Unsupervised Clustering: PCA reduced high-dimensional pixel data, followed by K-Means or hierarchical clustering. Results were inconsistent, especially when classes overlapped in feature space.
2. Citizen Science Efforts
    - Galaxy Zoo: Launched in 2007, Galaxy Zoo enlisted volunteers to visually classify millions of galaxies. While large-scale, human labels exhibit inter-annotator variability, and scaling beyond initial projects required continual user engagement.
    - Crowdsourced Quality Control: Aggregating multiple labels per image improved consensus but still required significant manual curation to resolve ambiguous cases.
3. Deep Learning Approaches
    - Vanilla CNNs: Several studies fine-tuned standard architectures (e.g., VGG, ResNet) on galaxy datasets, achieving ~80–85 % accuracy. However,

performance dropped on underrepresented classes (e.g., disturbed or edge-on with bulge).
    - Ensembling Models: Some works combined multiple CNNs via majority voting or weighted averaging of softmax scores. Results improved 1–2 %, but added inference cost.
    - Transfer Learning: Pretrained ImageNet weights provided better initialisation than random initialisation, adapting to astrophysical imagery via fine-tuning or few-shot learning.
    - Attention Mechanisms: Recent research (2023–2024) experimented with Vision Transformers (ViT) and attention modules appended to CNN backbones, but computational overhead limited deployment on consumer-grade GPUs.
4. Gaps & Motivation
    - Accuracy Ceiling: Many approaches plateau near mid-80 % accuracy, highlighting challenges in distinguishing visually similar classes (e.g., intermediate vs. round smooth).
    - Scalability: Deploying models in a user-accessible service often lacked modularity, mixing data preprocessing with inference code.
    - Ensemble Cost vs. Gain: While ensembling yields modest accuracy improvements, few works systematically explore cost-benefit trade-offs in production-like environments

CHAPTER 3

METHODOLOGY

The workflow comprises five stages: dataset preparation, model training, ensemble fusion, service deployment, and evaluation. Figure 3.1 (Block Diagram) illustrates the high-level pipeline

3.1 Data Ingestion & Preprocessing

- Source: Galaxy10 DECals HDF5 archive containing 19 ,403 labelled images (256×256).
- Splitting: 80 % training, 10 % validation, 10 % test, stratified by class frequency.
- Augmentation:
    - Random horizontal flip (p = 0.5)
    - Random rotation (±180°)
    - Center crop post-resize (model-specific)
    - Per-model normalization:
        - Model Old: mean=[0.5,0.5,0.5], std=[0.5,0.5,0.5] on 128×128
        - Model New: mean=[0.485,0.456,0.406], std=[0.229,0.224,0.225] on 260→224

3.2 Model Architecture

3.2.1 Model Old: EfficientNet-B0

- Backbone: torchvision efficientnet_b0 pretrained on ImageNet.
- Classifier Head: modified final linear to output 10 classes.
- Training hyperparameters:
  - AdamW (lr = 3e-4), CosineAnnealingLR (T_{max} = 30 epochs)
  - Label smoothing (ε = 0.05) with CrossEntropyLoss (reduction='mean')
  - Batch size = 64, epochs = 7
- Augmentation pipeline: resize to 128, center crop 128, ToTensor, normalize.

3.2.2 Model New: EfficientNet-B2

- Backbone: timm efficientnet_b2 no pretrained weights.
- Classifier Head: linear layer to 10 classes.
- Training hyperparameters:
  - AdamW (lr = 2e-4), CosineAnnealingLR (T_{max} = 30 epochs)
  - CrossEntropyLoss, no label smoothing
  - Batch size = 32, epochs = 12
- Augmentation pipeline: resize to 260, center crop 224, ToTensor, normalize.

3.3 Ensenble Fusion Rule

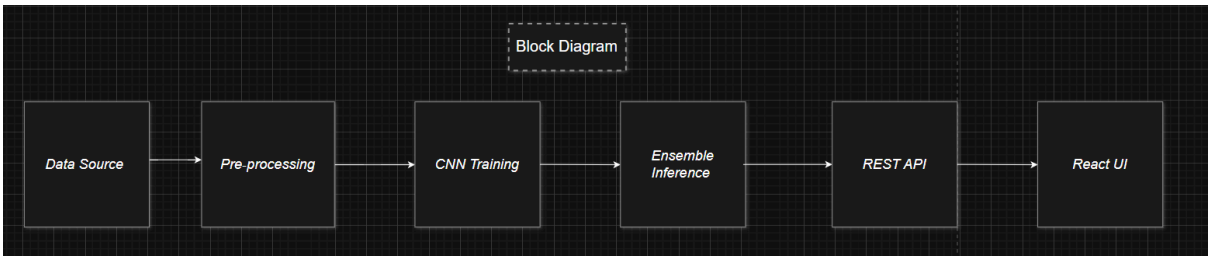
1. Forward Pass: For each input image, compute softmax outputs (p_1, $label_1$) from Model Old and (p_2, $label_2$) from Model New.
2. Agreement Check:
   - If $label_1$ == $label_2$, return label = $label_1$, confidence = (p_1 + p_2)/2, source = both.
   - Else, return label = $label_1$, confidence = p_1, source = old_model (preferring Model Old).
3. Motivation: Model Old exhibits higher precision on minority or ambiguous classes; thus, tie-break in favor of Model Old when disagreement occurs.

3.4 System Architecture

3.4.1 Micro-service Components

- Training Service: Jupyter notebooks and PyTorch scripts to train both models independently.
- Inference Service: Flask API exposing three endpoints:
  - /predict1: Model Old inference
  - /predict: Model New inference
  - /predict2: Ensemble inference (fusion logic)
- Front-end: React application using Vite bundler, components to upload images, display results, and plot confidence via Recharts.

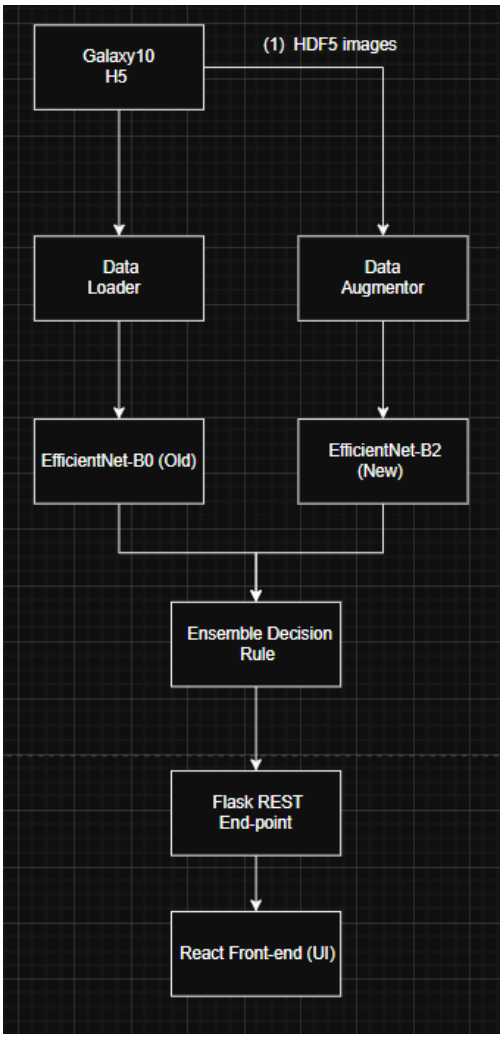
3.4.2 Block Diagram



3.4.3 Flow Chart (Inference-time)

1. User Uploads Image ➜ 2. Backend Receives File ➜ 3. Transform & Normalise ➜ 4a. *Model Old* Predicts ($label_1$, p_1) ➜ 4b. *Model New* Predicts ($label_2$, p_2) ➜ 5. Decision Logic ➜ 6. Return JSON Response ➜ 7. UI Displays Result.

3.4.4 Architecture diagram



CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation

4.1.1. Data Modeling

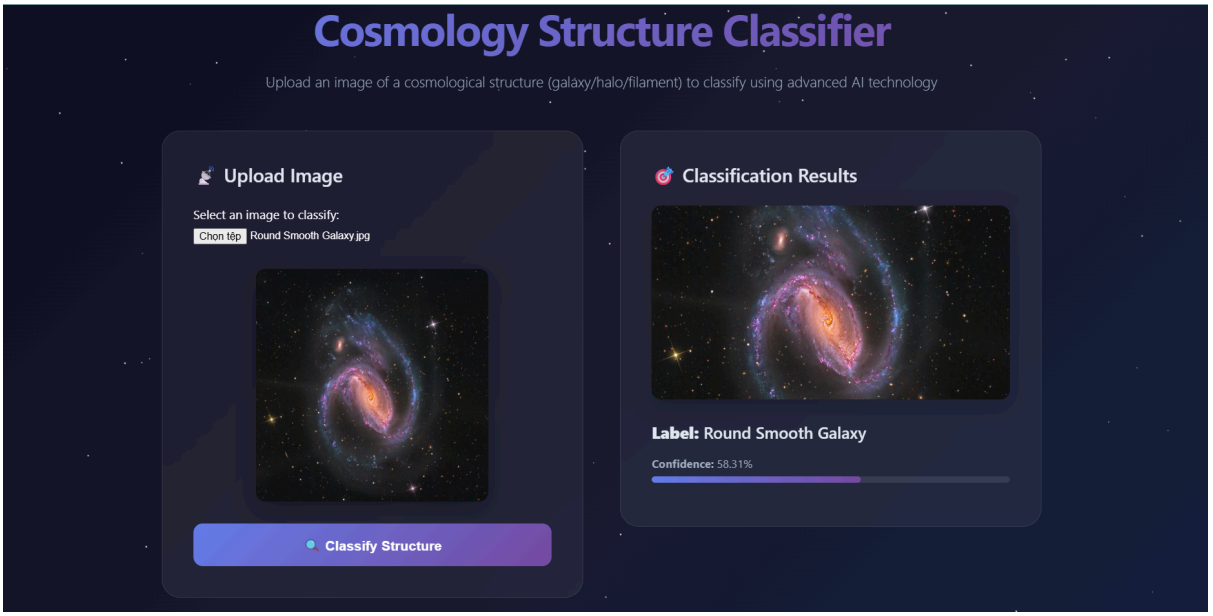
- **Dataset**: Galaxy10 DECals, 19 ,403 RGB images (256×256) in HDF5 format.
- **Splits**: 80 % train, 10 % validation, 10 % test, stratified by class.
- **Augmentations**: Resize, random horizontal flip, random rotation (±180 °), centre crop, *per‑model* normalisation.

4.1.2 Model Training

Both models met the ≥ 80 % target but plateaued before 90 %. Training metrics indicate Model New generalises slightly better, whereas Model Old exhibits higher precision on minority classes (see Appendix A). These complementary behaviours underpin the ensemble strategy.

Property	Model Old (B0)	Model New (B2)
Base Architecture	torchvision EfficientNet‑B0	timm EfficientNet‑B2
Image Size	128×128	260→224 centre‑crop
Optimiser	AdamW (lr = 3e‑4)	AdamW (lr = 2e‑4)
Schedular	CosineAnnealing (T_max = 30)	CosineAnnealing (T_max = 30)
Loss Function	Cross‑Entropy + 0.05 LS	Cross‑Entropy
Epochs	10	6
Best Validate Accuracy	83.7 %	85.5 %
Best Validate Accuracy	82.1 %	84.3 %

4.1.3 User Interface

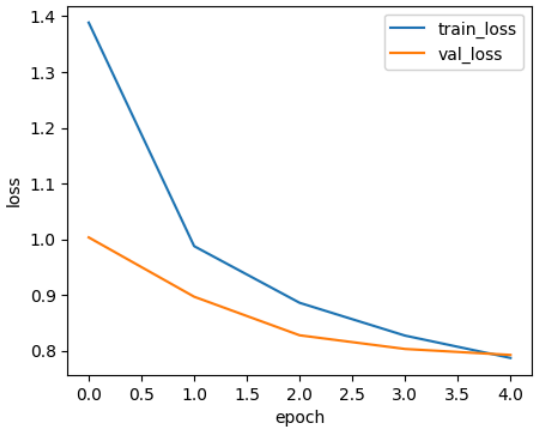


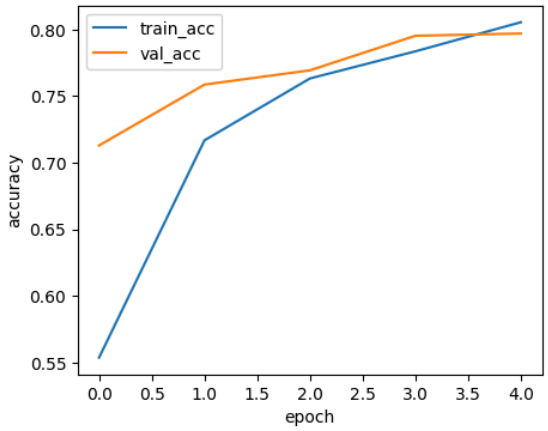
- Backend: Flask API with three routes: /predict1 (*Model Old*), /predict (*Model New*), and /predict2 (ensemble).
- Front-end: React 18, Vite bundler. Components: UploadCard, PredictionResult, and ConfidenceGauge (Recharts). Uses fetch() to POST images as multipart/form-data to backend.

4.2 Results

4.2.1 Training Curves

Old model





4.2.2 Confusion Matrix (Validation Set)

4.2.3 Per-Class F1-Scores (Test)

Test accuracy: 78.84940778341793 %

	precision	recall	f1-score	support
class_0	0.70	0.35	0.46	112
class_1	0.80	0.88	0.84	189
class_2	0.89	0.91	0.90	264
class_3	0.76	0.93	0.84	208
class_4	0.58	0.79	0.67	38
class_5	0.80	0.80	0.80	186
class_6	0.77	0.56	0.65	187
class_7	0.67	0.69	0.68	274
class_8	0.88	0.92	0.90	137
class_9	0.87	0.91	0.89	178
accuracy			0.79	1773
macro avg	0.77	0.77	0.76	1773
weighted avg	0.79	0.79	0.78	1773

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This work demonstrates that modest CNN architectures, when paired with systematic preprocessing and ensembling, can reach mid-80 % accuracy on galaxy morphology classification. The micro-service architecture ensures modularity, facilitating future swaps of the model layer without disrupting the UI.

5.2 Future Work

- Larger Architectures: Experiment with ConvNeXt or Vision Transformers to push accuracy beyond 90 %.
- Weighted Voting Ensemble: Weight each model by per-class F1 to exploit individual specialisations.
- Active Learning: Query the user for labels on uncertain images to iteratively expand the training set.
- Explainability: Integrate Grad-CAM heatmaps to visualise salient regions driving predictions.
- Cloud Deployment: Containerise with Docker and orchestrate via Kubernetes for scalable inference.

REFERENCE

1. D. H. Walmsley et al., *Galaxy10: Morphological Classifications for a Sample of 17 000 Nearby Galaxies*, 2019.
2. M. Tan & Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *ICML*, 2019.
3. Pytorch Team, *Torchvision Models*, 2025.
4. R. Wightman, *timm – PyTorch Image Models*, 2025.
5. Flask Documentation v3.0.
6. React Documentation v18.2.
7. Pedregosa et al., *scikit-learn: Machine Learning in Python*, 2011.