School of Computer Science
Faculty of Science and Engineering
University of Nottingham
Malaysia

UG FINAL YEAR DISSERTATION REPORT

Interpretable Seagull classification

Student's Name : Aravindh Palaniguru

Student Number : 20511833

Supervisor Name : Dr. Tomas Maul

Year : 2025

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE WITH ARTIFICIAL INTELLIGENCE (HONS)
THE UNIVERSITY OF NOTTINGHAM

University of Nottingham
UK | CHINA | MALAYSIA

INTERPRETABLE SEAGULL CLASSIFICATION

Submitted in May 2025, in partial fulfillment of the conditions of the award of the degrees B.Sc.

Aravindh Palaniguru
School of Computer Science
Faculty of Science and Engineering
University of Nottingham
Malaysia

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____ / _____ / _____

Acknowledgement

I would like to express my deepest gratitude to my Final Year Project supervisor, Dr Tomas Maul, for his exceptional guidance, and support throughout both semesters of my dissertation journey. His expertise in computer science and artificial intelligence was crucial to the successful completion of my dissertation, providing invaluable direction and insight at every stage of the project. This project has been the most intellectually stimulating and fulfilling work I have undertaken during my degree.

I am deeply thankful to Professor Chris Gibbins for sharing his ornithological expertise, providing his high-quality gull images without a second thought, and offering critical guidance throughout the project. His contribution was essential to ensuring the scientific validity and practical relevance of this work.

This project would not have been possible without the mentorship and encouragement I received, which made this challenging journey both rewarding and enlightening. Engaging with the topic of interpretable seagull classification has been a truly meaningful and enjoyable experience for me. Each stage of the project-from dataset preparation and model development to implementing interpretability techniques and analyzing subtle species differences-was intellectually stimulating and offered new learning opportunities. The collaborative environment, support from my supervisors, and the chance to work at the intersection of artificial intelligence and biodiversity made this journey especially memorable and fulfilling.

Abstract

This research addresses the challenge of fine-grained bird classification by developing an interpretable deep learning framework for distinguishing between two morphologically similar gull species: Slaty-backed Gull (Larus schistisagus) and Glaucous-winged Gull (Larus glaucescens). The study implements and evaluates multiple transfer learning models, with VGG-16 achieving the highest test accuracy of 95.74%. Grad-CAM interpretability techniques reveal the models' focus on wing and wingtip regions, aligning with ornithological knowledge. Quantitative analysis confirms statistically significant differences in wing intensity between species, with Slaty-backed Gulls exhibiting mean wing intensities significantly darker than Glaucous-winged Gulls. Feature extraction and unsupervised clustering validate these findings, achieving 95.6% classification accuracy using intensity-based features. This work demonstrates how deep learning interpretability can bridge artificial intelligence and biological taxonomy, providing both an effective identification tool and quantitative insights into morphological distinctions between closely related species.

Table of Contents

1 Introduction

Biodiversity is under unprecedented pressure due to climate change and human influence. The alarming rates at which species are disappearing indicate that the sixth mass extinction is underway [1]. Understanding what biodiversity we have and what we stand to lose is crucial for convincing decision-makers to take appropriate conservation action. The gaps in taxonomic knowledge and shortage of experts constitute what is known as the ”taxonomic impediment” [2], which hampers our ability to document and protect biodiversity effectively. Determining whether two populations can be consistently distinguished based on morphological traits remains essential for establishing taxonomic boundaries and designing appropriate conservation strategies.

Birds are frequently utilized to assess environmental quality due to their sensitivity to ecological changes and ease of observation during field studies. Researchers often rely on bird diversity as an indicator of the diversity within other species groups and the overall health of human environments. Examples include monitoring environmental changes through bird population shifts, tracking climate change via bird migration patterns, and evaluating biodiversity by counting bird species. Accurate identification of bird species is essential for detecting species diversity and conserving rare or endangered birds promoting scientific research and conservation efforts [3].

Among birds, the classification of gulls presents multiple challenges that make traditional identification methods problematic and inconsistent. Multiple confounding factors complicate identification [4], including **hybridization** where species can interbreed in overlapping ranges, creating intermediate forms; **age-related variations** as juvenile and immature gulls display less distinct patterns than adults; **environmental effects** such as feather bleaching from sun exposure, contamination, and wear that can alter appearance; **seasonal moulting** where gulls undergo plumage changes throughout the year, affecting diagnostic features; and **viewing conditions** where lighting, angle, and distance significantly impact observed coloration.

As noted by [5] in *The Gull Guide*,

> Gulls can be a challenging group of birds to identify. To the untrained eye, they all look alike, yet, at the same time, in the case of the large gulls, one could say that no two birds look the same.

[4] highlight the particular challenges with Glaucous-winged Gulls,

> Glaucous-winged Gulls also exhibit variably pigmented wingtips . . . these differences are often chalked up to individual variation, at least by this author, but they’re inconveniently found in several hybrid zones, creating potential for much confusion.
>
> The amount of variation here is disturbing because it is unmatched by any other gull species, and more so because it is not completely understood.

This project addresses the challenge of fine-grained analysis between two closely related gull species: the Slaty-backed Gull (*Larus schistisagus*) and the Glaucous-winged Gull (*Larus glaucescens*). These species display subtle and overlapping physical characteristics that make accurate identification highly complex.

There are many methods that can be used for classification. Manual identification to classify species requires per specimen analysis by expert taxonomists which is

time consuming. Automated taxon identification systems (ATIs) could both handle routine identifications and potentially assist in identifying new species. Traditional ATIs, however, have been limited by their reliance on hand-crafted features [6], are time-consuming hindering large-scale surveys, making them difficult to generalize across different taxonomic groups. While using machine learning techniques to solve the problem of fine-grained classification, as mentioned by [7], traditional feature extraction approaches rely on manually crafted features-such as edge detection, color histograms, keypoint matching, and visual word bag which often have limited expressive power and demand detailed manual annotations like bounding boxes and keypoints. The main limitation of these methods is the significant manual effort required for both selecting and extracting features.

Fine-grained image classification (FGIC), which focuses on identifying subtle differences between subclasses within the same category, has advanced rapidly over the past decade with the development of sophisticated deep neural network architectures. Deep learning approaches offer promising solutions to this taxonomic challenge through their ability to automatically learn discriminative features from large datasets[8].

Unlike traditional machine learning methods that rely on hand-engineered features, deep neural networks can detect complex patterns in high-dimensional data, making them well-suited for fine-grained visual classification tasks [6]. Features extracted through convolution are learned automatically by multilayer convolutional neural networks, offering the model greater adaptability to various tasks and datasets, with features possessing enhanced expressive and abstract capabilities. The benefit of convolutional feature extraction is its ability to perform feature extraction and classification within the same network, with the quality and quantity of features adjustable through the network's structure and parameters offering the model greater adaptability to various tasks and datasets [7]. Getting good quality results in traditional machine Learning models is dependent on how good the data is labelled, whereas deep Learning architectures don't necessarily require labelling, as Neural Networks are great at learning without guidelines [9].

For species identification, convolutional neural networks (CNNs) such as ResNet, Inception, and VGG have demonstrated exceptional capabilities [10] with accuracies of above 75%. Studies such as [11] who exceeded 97% in bird species classification tasks mentioned that deep learning is more effective than traditional machine learning algorithms in image recognition as the number of bird species increases achieving accuracy rates. [12] compared deep learning and traditional machine learning algorithms and achieved an accuracy of 94% tackle the challenge of classifying bird species with high visual similarity and subtle variations. These architectures automatically learn hierarchical feature representations—from low-level edges and textures to high-level semantic concepts—that capture the subtle morphological differences between closely related species.

Yet the fine-grained bird classification task has greater challenges even when using deep learning. Key issues include high intraclass variance, low inter-class variance, limited training data, intensity and pose variability, bird localization and background variability, occlusions, size variations, class imbalance [13], and limited samples[14] [3]. Additionally, deep learning requires extensive data and is prone to overfitting.

2 Literature Review

Deep Learning for Fine-Grained Image Classification

Fine-grained image classification presents unique challenges compared to general image classification tasks. Fine-grained classification "necessitates discrimination between semantic and instance levels, while considering the similarity and diversity among categories" [7]. This is particularly challenging in bird classification due to three key factors: high intra-class variance (birds of the same species in different postures), low inter-class variance (different species with only minor differences), and limited training data availability, especially for rare species[7].

Early approaches to fine-grained classification relied on fixed rectangular bounding boxes and part annotations to obtain visual differences, but these methods required extensive human annotation effort. Recent research has shifted toward weakly supervised approaches that only require image-level labels, developing localization subnetworks to identify critical parts followed by classification subnetwork [7]. These models facilitate learning while maintaining high accuracy without needing pre-selected boxes, making them more practical for real-world applications.

Transfer Learning for Image Classification

Deep learning, while powerful, comes with two major constraints: dependency on extensive labeled data and high training costs[15]. Transfer learning offers a solution to these limitations by enabling the reuse of knowledge obtained from a source task when training on a target task. In the context of deep learning, this approach is known as Deep Transfer Learning (DTL)[15].

Several studies have demonstrated the efficacy of transfer learning for bird species classification. A study on bird species identification using deep learning achieved accuracies of above 90% by leveraging pretrained CNN networks with a base model to encode images[16]. Research on bird species identification by [17] using modified deep transfer learning achieved 98.86% accuracy using the pretrained EfficientNetB5 model. The results with various pretrained models achieving high accuracy rates with few models exceeding 98% demonstrate that transfer learning approaches can achieve high performance even with limited training data.

Various pretrained models have been evaluated for bird classification tasks, including VGG16, ResNet, DenseNet, and EfficientNet architectures. Comparative studies have shown that while all these models can perform effectively, some consistently outperform others. For example, research on drones-birds classification found that "the accuracy and F-Score of ResNet18 exceeds 98% in all cases"[18], while another study on binary classification achieved the best classification accuracy of 99.49% with ResNet[19].

In a noteworthy study on bird classification analysis, researchers evaluated the comparative performance of MobileNetV2 and Inception-v3 classification models. The investigation employed four distinct methodologies: implementing Inception-v3 both with and without transfer learning, and similarly applying MobileNetV2 with and with-

out transfer learning techniques. The experimental results demonstrated that the MobileNetV2 architecture leveraging transfer learning capabilities achieved superior performance, reaching approximately 91.00% accuracy in classification tasks [20]. An experiment conducted by [21] conducted a comprehensive evaluation of different CNN architectures for identifying local bird species. With only 100 images per class before data augmentation high accuracies of above 90% were achieved.

Transfer learning addresses the primary challenges of deep learning: the need for large datasets and extensive computational resources. By leveraging pretrained models that have already learned general visual features from massive datasets, transfer learning enables the development of highly accurate classifiers with relatively domain-specific datasets. This is particularly valuable for this project, which focuses on distinguishing between two specific gull species with limited available data. [15] emphasizes that recent transfer learning methods aim to reduce training process time and cost, and the necessity of extensive training datasets. [11] using CNN models pretrained from ImageNet achieved above 90% accuracy in many CNN most that were tried for bird classification using transfer learning emphasize, ”when the sample data is small, transfer learning can help the deep neural network classifier to improve classification accuracy.” This makes transfer learning an ideal approach for specialized tasks like distinguishing between closely related gull species.

Interpretability Techniques for Deep Learning Models

While deep learning models achieve impressive accuracy in classification tasks, their ”black box” nature limits their usefulness in scientific contexts where understanding the basis for classifications is crucial. Interpretability techniques address this limitation by providing insights into model decision-making processes, making them essential tools for applications where transparency is as important as accuracy.

Gradient-weighted Class Activation Mapping (Grad-CAM) has emerged as a particularly valuable technique for visualizing regions of images that influence classification decisions. Selvaraju et al. [22] introduced this technique as a generalization of CAM that ”uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map highlighting important regions in the image.” Unlike earlier methods, Grad-CAM requires no architectural changes and can be applied to almost any CNN-based model. Grad-CAM ”uses the gradients of each target that flows into the least convolutional layer to produce a bearish localization map, highlighting important regions in the image for concept prediction” [23]. This approach enables validating model decisions against expert knowledge and potentially discover new insights about morphological features with regards to this project.

Selvaraju et al. [22] demonstrated that Guided Grad-CAM outperforms baseline approaches such as Guided Backpropagation and Deconvolution in generating class-discriminative visualizations. Their results show that Guided Grad-CAM not only produces high-resolution explanations but also clearly highlights the specific image regions that influence the model’s classification decision—a property that is vital for this project, where understanding the distinguishing features between classes is essential. The visualizations allows us to see the exact areas that contributed to the class decision, thereby increasing transparency and trust in the classifier.

3 Aims and Objectives

This dissertation aims to develop and evaluate deep learning models for the automated classification of the Slaty-backed Gull and the Glaucous-winged Gull, to use them to identify the most influential regions that distinguish these two species and analyse them to extract statistical features from these regions and assess their significance and validity in classification.

3.1 Aims

1. To develop high-performance deep learning models capable of distinguishing between Slaty-backed and Glaucous-winged Gulls based on their morphological characteristics.
2. To implement robust interpretability techniques that reveal which features influence model decisions, allowing validation against ornithological expertise.
3. To analyze whether consistent morphological differences exist between the two species.
4. To identify key discriminative regions and perform analyses to get statistical features.
5. To find out if the results of the analysis are statistically significant.
6. To test whether the features can be used to identify the species of a gull with high confidence.

3.1.1 Objectives

The project was carried out in three phases:

1. Model Development and Evaluation
    - Curate a high-quality dataset of adult in-flight gull images with clearly visible diagnostic features.
    - Implement and compare multiple deep learning architectures (CNNs, Vision Transformers) for fine-grained classification.
    - Evaluate models on unseen test sets.
2. Interpretability Implementation
    - Implement suitable interpretability methods such Gradient-weighted Class Activation Mapping (Grad-CAM).
    - Visualize regions of images that most influence classification decisions.
    - Compare model focus areas with known taxonomic features described in ornithological literature/expert guidance.
3. Features Analyses and Validation
    - Perform quantitative analysis of image regions highlighted by interpretability techniques.
    - Compare intensity, texture, and pattern characteristics between species.
    - Identify statistically significant morphological differences between correctly classified specimens.
    - Validate the features that are significantly different to check if they can be used for classification using clustering.

4 Methodology

4.1 Google Colab Platform

Google Colab was selected as the primary platform for developing and training deep learning models for its cloud-based GPU acceleration, integration with Google Drive, and pre-installed libraries, facilitating efficient model development. Despite its advantages, Google Colab presented a few challenges. The platform frequently disconnected during training sessions, interrupting the model training process before completing all epochs. As noted by [24], while Colab provides robust GPU resources that can match dedicated servers for certain tasks, these free resources "are far from enough to solve demanding real-world problems and are not scalable."

The relatively small size of our dataset helped minimize resource demands. Checkpoint saving was implemented throughout the training process, allowing training to resume from the last saved state if disconnections were encountered. This approach ensured that progress wasn't lost when disconnections occurred, though it introduced some workflow inefficiencies.

4.1.1 Python and PyTorch Framework

The implementation utilized Python and PyTorch, chosen for their extensive machine learning ecosystem. The implementation utilized Python with PyTorch, capitalizing on its dynamic computation graphs for streamlined debugging and flexible model customization. The framework's efficient DataLoader pipelines enabled effective batch processing, while its native gradient support facilitated the integration of Grad-CAM for improved model interpretability.

4.2 Dataset Preparation and Refinement

The dataset preparation followed a three-stage iterative refinement process, each addressing specific challenges identified during model development.

4.2.1 Stage 1: Initial Dataset Collection

The initial dataset was collected from public repositories including eBird and iNaturalist, comprising 451 images of Glaucous-winged Gulls and 486 images of Slaty-backed Gulls. This dataset included gulls of various ages (juveniles and adults) in different postures (sitting, standing, and flying). Initial model testing on this dataset yielded poor performance (below 60% accuracy), highlighting the need for dataset refinement.

4.2.2 Stage 2: Refined Dataset - Focus on Adult In-flight Images

Consultation with Professor Gibbins, an ornithological expert, revealed that wing and wingtip patterns were the most reliable distinguishing features between these species. These patterns are most visible in flight. Juvenile images were removed due to their less defined wingtip features and differing plumage. Consequently, the dataset was refined to focus exclusively on adult in-flight images, resulting in a curated collection

of 124 Glaucous-winged Gull images and 127 Slaty-backed Gull images. This targeted approach significantly improved model performance, with accuracy increasing to around 70%.

By focusing specifically on adult in-flight images where wingtip patterns are most visible, this project addresses the core taxonomic question while minimizing confounding variables. The resulting interpretable classification system aims to provide both a practical identification tool and a scientific instrument for exploring morphological variation within and between these closely related species.

4.2.3 Stage 3: High-Quality Dataset

To further enhance classification performance, 640 high-resolution images of in-flight Slaty-backed Gulls were obtained from Professor Gibbins. More Glaucous-winged Gull images were collected and curated with expert guidance, reducing it to 135 high-quality images that clearly displayed critical wingtip features. Images showing birds in moulting stages, juveniles, or unclear wingtip patterns were removed.

For comparative analysis, an unrefined dataset containing 632 adult in-flight Glaucous-winged Gulls and 640 high-quality Slaty-backed Gull images was also tested which resulted in around 60% model accuracy.

Test Dataset

All well-performing models were subsequently evaluated using a test dataset comprising unseen images during training, sourced from books, online, and from industry expert Professor Gibbins in order to obtain a precise assessment of model accuracy. In the final evaluation, the test set consisted of 14 images of Glaucous-winged Gulls and 33 images of Slaty-backed Gulls. Although this test set was imbalanced, the results were not adversely affected, as the models did not exhibit bias toward either class. This is further supported by the individual class metrics presented in the Section 5, which demonstrate consistent performance across both classes.

4.3 Iterative Development Methodology and Debugging

Initial implementations using ResNet50 with unrefined Stage 1 dataset yielded poor results (test accuracies below 60%), indicating fundamental issues in either data quality or model implementation. To systematically address these challenges and improve performance for subsequent transfer learning approaches, a methodical debugging framework was employed following best practices outlined by [25].

4.3.1 Pipeline Validation and Early Debugging

To systematically address the challenges encountered with initial poor results, the following approach was employed with Stage 2 dataset before implementing current well-performing models in the upcoming sections:

- **Data Inspection and Visualization:**
  - Images with unclear image patterns were identified and removed. With an imbalanced and a small dataset that we had, it was important not to provide unclear images to the model to prevent it from learning incorrect features

although the resulting dataset was small.
  - Augmentation visualization confirmed that features critical for classification (particularly wingtip patterns) remained visible after transformation
- **Pipeline Verification with Simple Models:**
  - A simple, lightweight Custom CNN was implemented as an initial baseline before advancing to complex architectures
  - This simplified model validated data loading procedures, augmentation effectiveness, and basic training operations
- **Single-Batch Overfitting Test:**
  - To verify gradient flow and learning capability, a single batch was deliberately overfitted with the simple CNN implemented
  - Training loss reduction from 0.7072 (Epoch 1) to 0.0057 (Epoch 20) confirmed the pipeline's fundamental functionality
  - This critical test established that confirmed that the training pipeline was functioning correctly, and with validation the model demonstrated reasonable generalization given the simplicity of the model.
- **Controlled Experimentation:**
  - Random seeds were fixed across all implementations (set to 42) to ensure reproducibility
  - This approach eliminated training variability as a confounding factor when comparing architectural modifications
  - Systematic adjustments to hyperparameters could be evaluated with confidence that performance differences were attributable to the specific changes rather than random initialization

After establishing a robust development pipeline and refining the dataset, the transfer learning implementations described in the following sections achieved significantly improved results, with test accuracies exceeding 90% for the best-performing models.

4.4 Transfer Learning Approach

Transfer learning was employed in the implementation to leverage the robust feature extraction capabilities of pre-trained models on ImageNet. This approach aligns with best practices in fine-grained classification tasks, where lower-level features learned from diverse datasets can be effectively repurposed for specialized domains with limited data. The pre-training on ImageNet's 1.2 million images across 1,000 classes provides the model with a strong foundation for recognizing a wide range of visual patterns [26], which can then be fine-tuned for our specific classification task despite class imbalance challenges.

The effectiveness of transfer learning was evident in the rapid convergence and high accuracy achieved even with our relatively limited dataset, demonstrating the potential of this approach for specialized classification tasks with significant class imbalance.

4.4.1 Common Implementation Strategy

All models except for the custom CNN utilized transfer learning to leverage knowledge from pre-trained networks. All the models mentioned in this section used the Stage 3

dataset. The transfer learning strategy included:

- Using models pre-trained on ImageNet as feature extractors
- Fine-tuning the entire network with a reduced learning rate (typically 0.0001 to 0.001) across models.
- Replacing the final classification layer to output binary predictions (2 classes)
- Implementing a dropout layer (dropout=0.4) before final classification to prevent overfitting

This approach follows the established pattern that features learned in early layers of convolutional networks are more general and transferable, while later layers become more task-specific.

4.4.2 Data Preparation and Augmentation

Data augmentation was crucial to address the limited dataset size and class imbalance issues. Following best practices from [27], multiple augmentation techniques were applied consistently across all models:

- **Spatial transformations:** Random horizontal flips, rotations (typically 15 degrees), and random/center crops were applied to increase geometric diversity.
- **Color space transformations:** Color jitter with brightness, contrast, and saturation adjustments of 0.2 magnitude was applied to make models robust to illumination variations.
- **Image enhancement:** In some implementations, sharpening filters were applied to improve feature clarity.
- **Normalization:** All images were normalized to match pre-trained model expectations [28].

The augmentation strategy was deliberately more aggressive for the training set compared to validation and test sets, where only resizing, optional cropping, and normalization were applied to maintain evaluation consistency. These techniques enhance model robustness to natural variations in image appearance, reducing overfitting and improving generalization capability [27].

4.4.3 Image Preprocessing

All images were preprocessed through a standardized pipeline:

Images were resized to match the architecture's expected input dimensions (224×224 pixels for most models, 299×299 pixels for Inception v3). Pixel values were normalized using ImageNet mean values [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225], ensuring input distributions aligned with those seen during pre-training.

4.4.4 Training Optimization Strategy

Parameter choices were based on literature, as early-stage hyperparameter tuning with Optuna did not significantly improve validation accuracy, and models were already achieving high accuracies without hyperparameter optimization. This approach prioritized established configurations over exhaustive tuning given the satisfactory baseline performance.

- **Optimizer:** AdamW optimizer with learning rates between 0.0001-0.001 and weight decay of 0.0005-0.001 as observed from [29] was used across implementations to provide adaptive learning with regularization. Learning rates less than, or upto the default pytorch value of 0.001 were set due to our small dataset in order to tackle potential underfitting.
- **Learning rate scheduling:** Adaptive learning rate scheduling using ReduceLROnPlateau was implemented across models, reducing learning rates by a factor of 0.1 (factor=0.1) when validation metrics plateaued for 3-5 consecutive epochs (patience=3-5). ReduceLROnPlateau with mode='max' was used to track validation accuracy improvements rather than minimizing loss.
- **Batch size:** The batch size was set to 16 for most of the models, striking a balance between computational efficiency and optimization stability. For a few trials batch size of 32 was also tested. Smaller batch sizes can increase gradient noise, which has been shown to act as an implicit regularizer that can improve generalization, particularly beneficial when working with limited training data [30, 31].
- **Early stopping:** The models were evaluated both with and without the application of early stopping. In experiments where early stopping was implemented, training was terminated if the validation accuracy did not improve for a predetermined number of epochs (patience = 3–5), or if the validation loss began to increase, thereby mitigating the risk of overfitting [32]. Yet the best performing model, which shows anti-overfitting evidence, was trained without early stopping.
- **Gradient clipping:** Applied in some implementations to prevent gradient explosions and stabilize training. Due to the small and imbalanced dataset, gradient clipping was implemented to prevent limited images from causing large weight updates.
- **Loss function:** Cross-entropy loss was used consistently as the optimization objective for the binary classification task.
- **Data splitting:** Train/validation split of 80%/20% was consistently used to provide reliable validation metrics while maximizing training data with random split due to our small dataset. Models with other trin/validation were also tested to note performance.
- **Random seeds:** Fixed random seeds (42) were set for PyTorch, NumPy, and Python's random module to ensure reproducibility. Controlling randomness is essential for reliable hyper-parameter tuning, performance assessment, and research reproducibility.

The combination of these techniques enabled effective learning despite the challenges of limited data and class imbalance, with our best model achieving significantly better performance than traditional machine learning approaches on the same dataset.

4.4.5 Regularization Techniques

Multiple regularization strategies were employed to handle the limited data size and class imbalance:

- **Dropout:** Layers with rates between 0.3-0.4 were consistently added before final classification layers to tackle overfitting due to our small dataset size [33].
- **Weight decay:** L2 regularization with weight decay values between 1e-4 and

1e-3 was applied across all models to prevent overfitting [34].

4.4.6 Addressing Class Imbalance

Our dataset exhibited significant class imbalance, which can degrade model performance by biasing predictions toward the majority class [35]. Working with limited datasets often introduces challenges related to class imbalance and overfitting. [36] conducted a comprehensive analysis of class imbalance in convolutional neural networks and found that oversampling (duplicating samples from minority classes) generally outperforms undersampling for deep learning models. Standard data augmentation techniques such as resizing, random horizontal and vertical flipping, rotation, affine translation, and color jittering can improve model generalization and robustness [37, 38]. Multiple complementary strategies were implemented on the best performing models that included VGG16, and ViT:

- **Class-Weighted Loss Function**
  - Implemented inverse frequency weighting [39]
  - Class weights calculation: $\text{class\_weights}[i] = \frac{\text{total\_samples}}{\text{num\_classes} \times \text{label\_counts}[i]}$
    PyTorch implementation: `CrossEntropyLoss` with class weights tensor
- **Weighted Random Sampling**
  - Balanced mini-batches using PyTorch's `WeightedRandomSampler`
  - Sample weights: $\text{samples\_weights} = \text{class\_weights}[\text{label}]$
  - Oversamples minority class and undersamples majority class [36]
  - Uses replacement sampling for effective batch balancing
- **Class-Specific Data Augmentation** [40] employed extensive data augmentation techniques for a fine-Grained Bird Classification. They demonstrated that such augmentations were particularly effective for classes with fewer samples, improving overall accuracy significantly by 7.99%.
  - Aggressive minority class augmentation [41]
  - Minority class transformations include:
    * 30° random rotations
    * Strong color jitter (brightness/contrast/saturation=0.3)
    * Random resized crops (scale=0.7-1.0)
    * Horizontal flips

    Standard augmentation for majority class (15° rotations, milder parameters)

Despite exploring various advanced techniques to mitigate class imbalance, such as oversampling, undersampling, and algorithmic adjustments, the most effective results were achieved using straightforward methods implemented with commonly adopted strategies mwntioned in Section 4.4.1.

4.4.7 Evaluation Strategy

Model performance was systematically evaluated using:

- **Validation accuracy:** Used during training to select optimal model checkpoints and trigger early stopping or learning rate adjustments.
- **Test accuracy:** The final evaluation metric was the accuracy on the unseen test set, which served to assess the generalization performance of the models. In

addition to evaluating test accuracy, Grad-CAM visualizations were generated using the test set to further aid in the selection of the optimal model and make sure that the model learned relevant features.

- **Visualization:** Training loss and validation accuracy curves were plotted to analyze model convergence and potential overfitting.
- **Checkpointing:** Best-performing models based on validation accuracy were saved for later evaluation on test dataset.

4.4.8 Model Checkpointing and Evaluation

Our implementation includes a robust evaluation framework with model checkpointing based on validation accuracy. This ensures that we preserve the best-performing model configuration throughout the training process. Models were in most cases trained for 20 epochs with early stopping implicitly implemented through best model saving in a few trials. Performance is evaluated using accuracy on both validation and test sets, providing a comprehensive assessment of model generalization.

This approach is well-aligned with findings by [42], who demonstrate that transfer learning with pre-trained models on bird classification tasks enables rapid convergence-often requiring as few as 2 to 10 epochs to achieve optimal accuracy in stark contrast to the 40–50 epochs typically needed for training models from scratch. This efficiency not only accelerates the experimental workflow but also reduces computational demands, making it feasible to conduct extensive model comparisons and hyperparameter tuning.

4.5 Model Architectures and Specific Implementations

4.5.1 VGG-16 Architecture

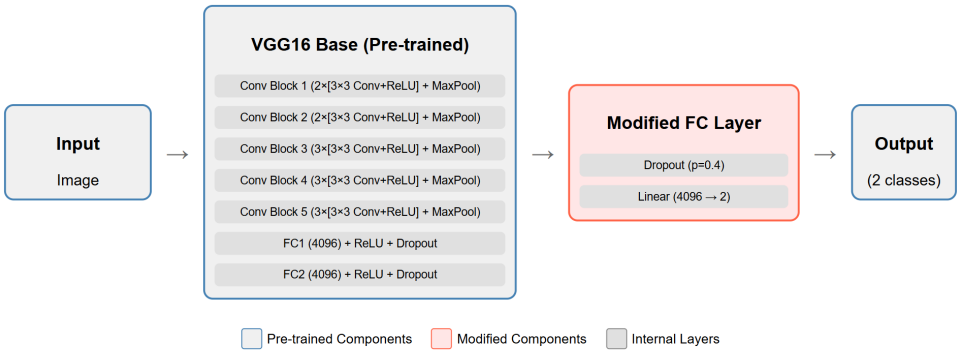


Figure 4.1: Architecture of the VGG model

Theoretical Foundation

VGG-16, developed by Simonyan and Zisserman (2014) [43] at the Visual Geometry Group (VGG) at Oxford, is a 16 layer CNN with 13 convolutional layers followed by 3 fully connected layers known for its deep stack of 3×3 convolutional filters followed by max pooling layers. With approximately 138 million parameters, VGG-16 provides

a strong foundation for feature extraction in computer vision tasks although it is computationally exprensive compared to other architectures. Pre-trained on the ImageNet dataset containing over 1.2 million images across 1,000 classes, VGG-16 offers robust initialization weights that facilitate effective knowledge transfer to domain-specific tasks with limited training data.

VGG-16 has demonstrated superior performance in fine-grained classification tasks compared to conventional techniques. Recent studies show that VGG-16 with logistic regression achieved 97.14% accuracy on specialized datasets like Leaf12, significantly outperforming traditional approaches that combined color channel statistics, texture features, and classic classifiers which only reached 82.38% accuracy [44]. For our specific task of gull species classification, the hierarchical feature representation capabilities of VGG-16 proved particularly effective at capturing the subtle differences in wing patterns and morphological features that distinguish between the target species.

By modifying the final classification layer to match the number of output classes, the VGG16 model can retain its ability to recognize general features and adapt to the unique visual characteristics of different bird species. This approach mirrored in my implementation, aligns with successful methodologies in avian species classification using VGG-16 as demonstrated by Brown et al. (2018), where fine-tuning the architecture by modifying the final classification layer enabled the model to retain general feature recognition capabilities while adapting to species-specific visual characteristics [45].

Model Adaptation for Fine-Grained Classification

The VGG architecture followed the common methodology approach mentioned in Section 4.4.1 with a dropout and FC layer mapping from the original 4096 features to 2 output classes replacing the original 1000-class classifier.

4.5.2 Vision Transformer (ViT) Architecture

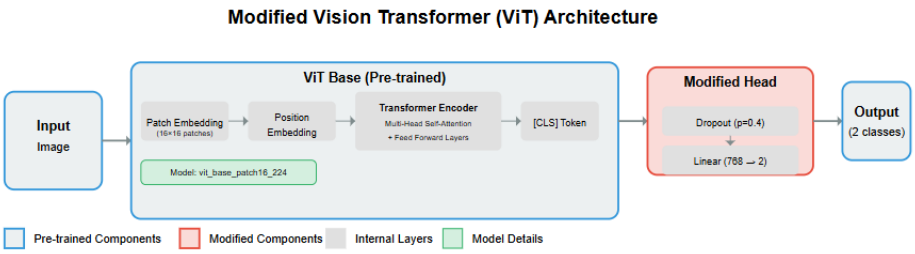


Figure 4.2: Architecture of the Vision Transformer (ViT) model

4.5.3 ViT for Fine-Grained Classification

Vision Transformers (ViT) have emerged as powerful alternatives to convolutional neural networks for visual recognition tasks. First introduced by Dosovitskiy et al. [46], ViTs process images as sequences of fixed-size patches, applying transformer-based self-attention mechanisms to model global relationships between image regions. This architecture enables the capture of long-range dependencies within images, making

it particularly suitable for fine-grained classification tasks where subtle distinctions between similar classes may depend on relationships between distant image features.

Vision Transformer Implementation

For our primary approach, a Vision Transformer was implemented using transfer learning from a pre-trained model with base architecture 'vit_base_patch16_224' pre-trained on ImageNet from the TIMM library [47]. The ViT architecture followed the common methodology approach mentioned in Section 4.4.1. The model architecture preserves the core self-attention mechanism of ViT while adapting the final classification layer for our specific binary classification task.

Alternative ViT Implementations

In addition to our primary implementation, we explored two attention-enhanced architectures for compatibility with Grad-CAM:

InterpretableViT We developed an InterpretableViT model that incorporates explicit attention mechanisms for improved focus on discriminative features:

- Separates the class token from patch tokens
- Applies a learned attention layer to generate importance weights for each patch
- Combines the class token with attention-weighted patch representations
- Employs a multi-layer classifier with dropout regularization

By separating the class token from patch tokens and implementing an explicit attention mechanism, the model facilitates more effective application of Grad-CAM, allowing for visualization of discriminative image regions contributing to classification decisions.

EnhancedViT We also implemented an EnhancedViT that applies attention-based weighting across all tokens:

- Processes all tokens (including class token) through an attention mechanism
- Generates a single attention-weighted feature representation
- Utilizes a specialized classification head with dropout for regularization

4.5.4 Inception v3 Architecture

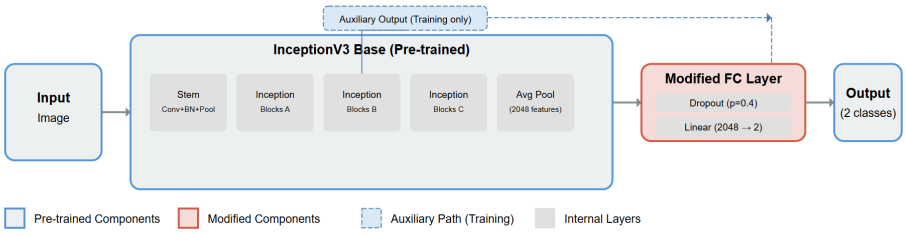


Figure 4.3: Architecture of the Inception v3 model

Theoretical Background

Inception v3 [48] employs a modular design centered around *Inception modules* that process input tensors through multiple parallel convolution pathways with varying receptive fields. This design efficiently captures multi-scale features simultaneously, making it particularly effective for fine-grained classification tasks requiring detection of subtle morphological differences.

The key architectural innovations in Inception v3 that informed our implementation include:

- Factorized convolutions that reduce computational complexity while maintaining representational power
- Auxiliary classifiers that inject additional gradient signals to mitigate vanishing gradients
- Batch normalization for training stability and faster convergence

Model-Specific Implementation Details

Our implementation adapted the pre-trained Inception v3 model for binary classification with auxiliary outputs. During training, we utilized Inception v3's auxiliary classifier to provide an additional gradient path during backpropagation, which is a distinctive feature of this architecture not found in our other implementations. Mixed precision training with torch.amp.GradScaler() was used to improve computational efficiency without sacrificing accuracy.

4.5.5 Residual Network (ResNet-50) Architecture

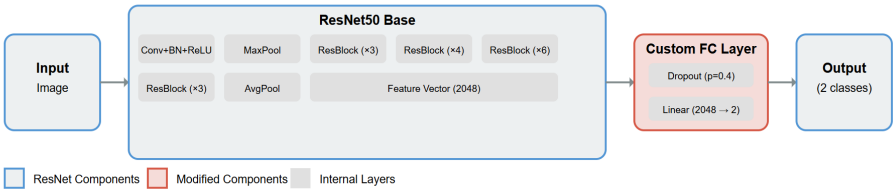


Figure 4.4: Architecture of the ResNet-50 model

Theoretical Background

ResNet-50 [49] addresses the degradation problem in deep neural networks through the introduction of residual connections (skip connections). These connections enable the network to learn residual mappings rather than direct mappings, allowing for substantially deeper architectures without suffering from vanishing gradients. The residual block can be formulated as:

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{4.1}$$

where $\mathcal{F}$ represents the residual mapping to be learned and x is the identity connection. This formulation makes optimization easier as the network only needs to learn the residual with respect to the identity mapping.

The ResNet architecture followed the common methodology approach mentioned in Section 4.4.1.

4.5.6 Custom CNN with Squeeze-and-Excitation Blocks

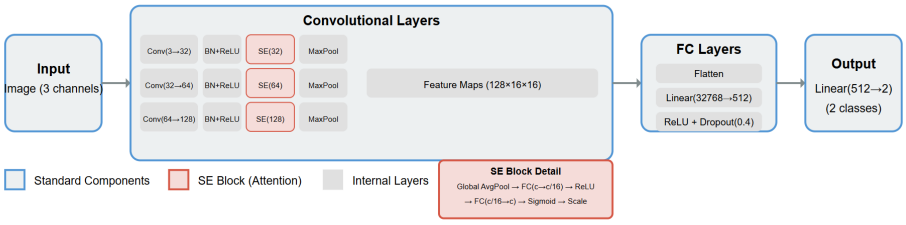


Figure 4.5: Architecture of the Custom CNN with Squeeze-and-Excitation Blocks

Architectural Design

To address the challenges of limited data and class imbalance in fine-grained classification, we developed a lightweight custom CNN architecture incorporating attention mechanisms. Our approach employs Squeeze-and-Excitation (SE) blocks, which enhance feature representation by modeling channel interdependencies through an attention mechanism. The SE block, as introduced by Hu et al. [50], adaptively recalibrates channel-wise feature responses to emphasize informative features while suppressing less useful ones.

The architecture consists of three convolutional blocks, each followed by batch normalization, ReLU activation, and an SE block. The SE block performs two key operations:

- **Squeeze**: Global average pooling across spatial dimensions to generate channel-wise statistics
- **Excitation**: A fully connected layer that produces modulation weights for each channel

This channel-wise attention mechanism has been shown to improve model performance with minimal computational overhead [50]. The SE blocks in our implementation use a reduction ratio of 16, balancing parameter efficiency and representational power.

Custom CNN-Specific Training Approach

Unlike the transfer learning approaches used with pre-trained models, our custom CNN was trained from scratch with some specific optimization strategies:

- **Cosine Annealing scheduler**: Our learning rate schedule follows a cosine annealing pattern with a period of 20 epochs, allowing the learning rate to oscillate and potentially escape local minima.
- **Batch normalization:** Used in custom CNN implementations to stabilize learning and improve convergence [51].
- **Specialized augmentation**: The custom CNN particularly benefited from more aggressive data augmentation strategies to compensate for the lack of pre-trained weights, including stronger rotations and more extensive color jittering than used

with the transfer learning models.

4.6 Model Interpretability Methodologies

Deep learning models, particularly Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), are often criticized for their lack of transparency, functioning as ”black boxes” wherein the decision-making process remains opaque to human observers. For critical applications like wildlife species classification, understanding how these models arrive at their predictions is essential for establishing trust and validating results. This section outlines the methodologies implemented to visualize and interpret the classification decisions of both the CNN architecture and Vision Transformer (ViT) models used in this study.

4.6.1 Gradient-weighted Class Activation Mapping (Grad-CAM)

Gradient-weighted Class Activation Mapping (Grad-CAM) is a widely adopted visualization technique that produces visual explanations for decisions made by CNN-based models without requiring architectural changes or retraining [22]. It generates coarse localization maps highlighting the regions in the input image that significantly influenced the model's prediction for a specific class.

Theoretical Foundation

Grad-CAM extends the Class Activation Mapping (CAM) approach [52] by utilizing the gradient information flowing into the final convolutional layer of a CNN. Unlike CAM, which requires modifications to the network architecture and retraining, Grad-CAM can be applied to any CNN-based architecture without architectural changes, making it more versatile.

The fundamental principle behind Grad-CAM is that the final convolutional layer in a CNN retains spatial information while encoding high-level semantics. By analyzing how the gradients of a specific class score flow into this layer, Grad-CAM can identify the regions in the input image that are most influential for the prediction.

The ReLU function is applied to the weighted combination of feature maps to focus only on features that have a positive influence on the class of interest, effectively eliminating features that suppress the class.

Methodology for CNN models

In this study, Grad-CAM was implemented for the VGG16 model by targeting the final convolutional layer (features[-1]). The implementation involves several key steps:

1. **Hook Registration**: Forward and backward hooks are registered on the target layer to capture activations during the forward pass and gradients during the backward pass.
2. **Forward Pass**: The input image is passed through the network to obtain the model's prediction.
3. **Backpropagation**: The gradient of the score for the target class (either the predicted class or a specified class) with respect to the feature maps of the target layer is computed through backpropagation.

4. **Global Average Pooling**: These gradients undergo global average pooling to obtain weights indicating the importance of each channel for the target class.
5. **Weighted Combination**: The weights are applied to the activations of the target layer to create a weighted combination of feature maps.
6. **ReLU Application**: A ReLU function is applied to the weighted combination to focus only on features that have a positive influence on the class of interest. (NumPy's maximum function is used to retain only positive values.)
7. **Normalization**: The resulting heatmap is normalized to the range [0, 1] for consistent visualization.
8. **Visualization**: The heatmap is resized to match the input image dimensions and overlaid on the original image using a colormap (typically 'jet') to highlight regions the model focused on for its prediction.

Similar implementation was implemented for other models: Inceptionv3, ResNet50, and CustomCNN.

4.6.2 Attention Rollout for Vision Transformers

Vision Transformers process images differently from CNNs, using self-attention mechanisms rather than convolution operations to model relationships between image patches. Therefore, a different approach called Attention Rollout is used to visualize ViT decision-making [53].

Theoretical Foundation

The Attention Rollout method visualizes information flow through the layers of a Transformer model. In Vision Transformers, the input image is divided into fixed-size patches, and each patch is linearly embedded along with position embeddings. A special classification token ([CLS]) is added, and the sequence of embedded patches is processed through multiple layers of self-attention.

Attention Rollout computes a measure of how the [CLS] token attends to each image patch by propagating attention through all layers of the network. This provides insight into which parts of the image the model considers most relevant for classification.

Methodology for ViT

The implementation of Attention Rollout for the ViT model follows these steps:

1. **Attention Map Collection**: Forward hooks are registered on each transformer block to collect attention maps during the forward pass of an input image.
2. **QKV Processing**: For each attention head, the query (Q), key (K), and value (V) matrices are extracted and processed to compute the raw attention weights between different tokens.
3. **Head Averaging**: Attention weights from all heads in each layer are averaged to get a single attention map per transformer block.
4. **Attention Rollout Computation**: Starting with an identity matrix, attention maps from each layer are sequentially multiplied to account for how attention propagates through the entire network.
5. **CLS Token Attention Extraction**: The attention weights from the classification

([CLS]) token to each image patch are extracted, which indicates the importance of each patch for the final classification.
6. **Reshaping and Visualization**: These weights are reshaped to match the spatial dimensions of the input image (typically 14×14 for ViT-Base with patch size 16) and then upsampled to create a heatmap that can be overlaid on the original image.

While the implementation includes a mechanism for filtering low-attention connections through a discard ratio parameter, this feature was not utilized in the final analysis (set to 0.0), focusing instead on the complete attention distribution across all patches.

4.6.3 Grad-CAM for Vision Transformers

In addition to Attention Rollout, this study also implements Grad-CAM for Vision Transformers to provide a more direct comparison with the CNN-based visualizations. Two variants of Grad-CAM for ViT were implemented: Enhanced ViT and Interpretable ViT, each with specific architectural modifications to facilitate interpretation.

4.6.4 Enhanced ViT Implementation

The Enhanced ViT approach modifies the standard ViT architecture to facilitate Grad-CAM visualization:

- **Model Architecture:** The Enhanced ViT model utilizes the base ViT by replacing the original classification head with an identity layer, preserving the full token representation from the transformer blocks.
- **Attention Mechanism:** An attention layer computes scalar importance scores for all tokens (including both [CLS] and patch tokens), which are then normalized through a softmax operation to produce attention weights.
- **Feature Integration:** These attention weights are applied to the token features through a weighted sum operation, creating a single feature vector that represents the entire image with emphasis on the most important regions.
- **Gradient Recording:** The model registers hooks to capture token features during the forward pass and their gradients during backpropagation, essential for the Grad-CAM calculation.
- **Grad-CAM Computation:** For visualization, the class token is excluded, and only patch tokens (corresponding to image regions) are considered. The tokens and gradients are reshaped from the sequence format back to a 2D spatial grid to recover the image structure.
- **Activation Map Generation:** Following standard Grad-CAM procedure, channel-wise weights are computed from gradients and applied to token features to create the final activation map highlighting discriminative regions.

4.6.5 Interpretable ViT with Feature Concatenation

The Interpretable ViT variant employs a feature concatenation approach that enhances interpretability:

- **Feature Extraction:** The implementation first extracts features from the base ViT model, producing a sequence of token embeddings.
- **Token Separation:** After feature extraction, the [CLS] token and patch tokens

are explicitly separated for differential processing:

$$\text{cls\_token} = \text{tokens}[:, 0, :] \tag{4.2}$$

$$\text{patch\_tokens} = \text{tokens}[:, 1:, :] \tag{4.3}$$

- **Attention Weighting:** An attention layer computes importance weights specifically for the patch tokens, determining each patch's contribution to the final representation:

$$\text{attn\_scores} = \text{attention\_layer}(\text{patch\_tokens}) \tag{4.4}$$

$$\text{attn\_weights} = \text{softmax}(\text{attn\_scores}, \text{dim} = 1) \tag{4.5}$$

- **Weighted Representation:** The patch tokens are combined using these attention weights to form a single weighted feature vector:

$$\text{weighted\_patch} = \sum_i \text{attn\_weights}_i \cdot \text{patch\_tokens}_i \tag{4.6}$$

- **Feature Concatenation:** The [CLS] token (providing global context) and the weighted patch features (providing localized information) are concatenated to form a comprehensive representation:

$$\text{combined} = [\text{cls\_token}; \text{weighted\_patch}] \tag{4.7}$$

- **Classifier Architecture:** To accommodate this concatenated representation, the classifier's first layer accepts double the embedding dimension as input:

$$\text{classifier} = \text{LayerNorm}(\text{embed\_dim} \times 2) \rightarrow \text{Dropout} \rightarrow \text{Linear} \rightarrow \ldots \tag{4.8}$$

- **Grad-CAM Visualization:** During Grad-CAM computation, gradients flow back through this architecture, capturing how both global context and local features influence the classification decision.

Both approaches enable visualization of the ViT's decision-making process using gradient information, but with different mechanisms for integrating token features and attention weights. The Enhanced ViT applies attention across all tokens before classification, while the Interpretable ViT explicitly separates and recombines different types of features to provide a more structured view of the classification process.

By implementing these complementary interpretability techniques, this research provides insights into how different neural network architectures—traditional CNNs and modern Transformers—approach the same classification task and what are the distinguishing features between the two classes. The visualizations reveal different feature priorities and decision strategies that each architecture employs, contributing to a deeper understanding of model behavior.

4.7 Additional Interpretability Techniques

In addition to the Grad-CAM and attention rollout methods discussed in the main sections of this report, several other interpretability techniques were explored during this research.

4.7.1 DeepLIFT

DeepLIFT (Deep Learning Important FeaTures) [54] was implemented using the Captum library for the VGG16 and ViT models. This technique compares activations against a reference baseline (typically zeros) to determine feature importance:

- A zero baseline tensor is created as a reference point
- Attribution scores are calculated comparing actual activations to this baseline
- Channel-wise attributions are aggregated and normalized for visualization

While DeepLIFT provides theoretically grounded attributions, it was not adopted for the main analysis due to its limited adoption in the literature compared to prevalent techniques like Grad-CAM.

4.7.2 Saliency Maps

- Gradients of predictions with respect to input pixels are computed
- Absolute values of gradients are visualized as sensitivity maps

4.7.3 Integrated Gradients for Vision Transformers

Integrated Gradients was implemented for ViT using 50 interpolation steps:

- Path integrals are approximated between baseline and input images
- 50 interpolation steps were used for the approximation
- Attributions are processed to align with spatial image dimensions
- Results are normalized for consistent visualization

4.7.4 Guided Backpropagation for ViT

- Modified gradient flow during backpropagation highlights positive influences
- Implementation uses Captum's GuidedBackprop with model-specific wrappers
- Results are post-processed to create single-channel visualizations

4.7.5 Justification for Grad-CAM

Grad-CAM was chosen as the primary interpretability method for this study due to its unique advantages over the alternatives:

- **Theoretical Robustness**: Grad-CAM passes sanity checks (e.g., parameter randomization tests) that methods like Guided Backpropagation, saliency maps and Guided Grad-CAM fail [55]. Findings of [55] implied that the saliency methods that failed their proposed tests were incapable of supporting tasks that require explanations that are faithful to the model or the data generating process.
- **Class-Discriminative**: Grad-CAM highlights regions relevant to specific predicted classes [22].
- **Architecture Flexibility**: Works consistently across CNNs (e.g., VGG16, ResNet) and adapted Vision Transformers.
- **Computational Efficiency**: Requires only a single backward pass per class, unlike path-based methods (e.g., Integrated Gradients).
- **Empirical Validation**: Grad-CAM is supported by extensive literature [22, 56] demonstrating its effectiveness for classification tasks.

4.8 Feature Analysis for Fine-Grained Gull Classification

4.8.1 Image Preprocessing and Region-Based Segmentation

A multi-stage approach was employed to extract and analyze region-specific features.

Image Selection

From the Stage 3 dataset, 200 high-resolution images (100 per species) were selected based on the following criteria:

- Correct classification by the best-performing VGG-16 model
- Prominent Grad-CAM activations in wing or wingtip regions
- Clear visibility of upperwing or wingtip in the image
- High resolution image

Manual Segmentation

Using Adobe Photoshop, segmentation masks with color-coded regions were created with a specific color for each type of region: wing, wingtip, and head.

Normalization

Min-max normalization[57] (0-255) was applied to grayscale conversions using OpenCV's `NORM_MINMAX`. This helped standardize intensity values and mitigate the effects of differing lighting conditions ensuring that pixel intensity values were scaled consistently across the dataset, providing a reliable foundation for subsequent region-based analysis.

Region Extraction

Using OpenCV's `inRange` for color-based thresholding, pixels that match each target colour were identified to create binary masks where the region of interest is set to 255 (white) and the rest to 0 (black). These binary masks were then applied to the original images using bitwise operations to extract the actual pixel values of the bird's regions—such as the wing, wingtip, head, and body—for subsequent analysis.

4.8.2 Local Binary Pattern Analysis

Local Binary Patterns (LBP) constitute a robust and computationally efficient texture analysis technique, originally proposed by Ojala et al. [58]. LBP characterizes local image texture by comparing each pixel's intensity with its surrounding neighbors, generating a binary code that summarizes the local structure. This descriptor is particularly advantageous for biological image analysis due to several key properties:

- Robustness to monotonic grayscale variations, enhancing resilience to changing illumination conditions.
- Rotation invariance when using specific LBP variants or derived features (1s and transition histograms)
- Ability to capture fine-scale micro-patterns potentially imperceptible to human vision.

To investigate subtle textural differences between Slaty-backed Gulls and Glaucous-winged Gulls, Local Binary Pattern (LBP) analysis was conducted on three anatomical regions: wing, wingtip, and head.

LBP Computation

For a given pixel at position (x_c, y_c) with intensity g_c, the LBP value is computed by comparing it with P neighbors at radius R using the following equation:

$$\text{LBP}_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p \tag{4.9}$$

where:

- g_p is the intensity of the neighbor pixel p
- $s(x)$ is the step function defined as:

$$s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{4.10}$$

$$\text{LBP}_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p, \quad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{4.11}$$

Two variants were implemented:

- **Default LBP**: Standard implementation without uniformity constraints
- **Uniform LBP**: Considers patterns with ≤ 2 bitwise transitions

Feature Extraction

Three feature histograms were derived from LBP codes:

1. **LBP Histogram**: Raw distribution of LBP codes
2. **Ones Histogram**: Frequency of patterns with n ones:

$$H_{\text{ones}}(i) = \sum_{j=0}^{N-1} [b(j) = i] \tag{4.12}$$

3. **Transitions Histogram**: Frequency of patterns with m transitions:

$$H_{\text{trans}}(i) = \sum_{j=0}^{N-1} [t(j) = i] \tag{4.13}$$

Texture properties were calculated from these histograms:

1. **Entropy**: Measures randomness in texture patterns

$$E = -\sum_i h(i) \log_2 h(i) \tag{4.14}$$

where $h(i)$ is the normalized histogram value at bin i

2. **Energy**: Measures uniformity of texture

$$U = \sum_i h(i)^2 \tag{4.15}$$

3. **Contrast**: Measures intensity variations

$$C = \sum_i (i - \mu)^2 h(i) \tag{4.16}$$

   where μ is the mean value of the histogram
4. **Homogeneity**: Measures closeness of distribution

$$H = \sum_i \frac{h(i)}{1 + |i - \mu|} \tag{4.17}$$

Comparative Analysis Implementation

To quantify the differences between species, the implementation calculates several distribution similarity metrics:

1. **KL Divergence**: A symmetric version of the Kullback-Leibler divergence that measures how one probability distribution diverges from another
2. **Earth Mover's Distance**: Measures the minimum "work" required to transform one histogram into another, considering the distance between bins
3. **Chi-Square Distance**: A statistical test that measures the difference between observed and expected frequency distributions
4. **Jensen-Shannon Distance**: A symmetric and smoothed version of the KL divergence with better numerical properties

Each metric captures different aspects of distribution similarity, providing a robust framework for comparing texture patterns between species.

This modular approach facilitates efficient processing of large image datasets and enables iterative refinement of the analysis parameters.

Discriminative Power Analysis

A systematic assessment of discriminative power for different texture features and regions is performed by calculating percentage differences between species for each texture property and region. This analysis identifies which features and regions exhibit the most significant differences between species. For each region and property combination, the implementation extracts the property value for each species, calculates the percentage difference, and ranks the region-property pairs by their discriminative power. This approach enables the identification of the most promising texture characteristics for species differentiation.

Dimensionality Reduction and Visualization

The implementation uses Principal Component Analysis (PCA) to visualize the high-dimensional LBP feature space in two dimensions. The process involves standardizing the feature data, applying PCA with 2 components, creating scatter plots colored by species, and calculating and displaying variance explained. This dimensionality reduction approach provides intuitive visualization of class separability and validates the discriminative power of the extracted features.

Implementation Workflow

The complete analysis pipeline consists of two main modules. The Feature Extraction Module loads original and segmentation images, extracts and processes each anatomical region, computes LBP features and abstract pattern features, and saves features to CSV files for further analysis. The Analysis Modules load extracted features, calculate advanced texture statistics, perform comparative analysis between species, generate visualizations and statistical reports, and identify the most discriminative features. This modular approach facilitates efficient processing of large image datasets and enables iterative refinement of the analysis parameters.

Validation Approach

The implementation includes a few validation mechanisms: normalized histograms to account for varying region sizes, multiple comparison metrics to ensure robust similarity assessment, and quantitative comparison metrics (KL divergence, percentage differences) to assess feature differences.

4.9 Wing and Wingtip Intensity Analysis

The interpretability analysis of our VGG model indicated a strong focus on wing and wingtip regions when differentiating between Slaty-backed Gulls and Glaucous-winged Gulls. This aligns with ornithological knowledge, as these species exhibit distinct differences in wing coloration patterns, particularly in the wing and wingtip regions. To quantitatively validate these differences and check whether the differences are significant, an in-depth image analysis was conducted focusing on these critical regions.

We employed a multi-stage approach to extract and analyze region-specific features:

4.9.1 Feature Extraction Pipeline

The Python pipeline executed these steps for each image:

1. **Region-Specific Analysis**: Calculated intensity statistics (mean, median, std, skewness, kurtosis, minimum and maximum values) per region
2. **Wingtip Characterization**:
   - Analyzed intensity distribution across 25 bins (10-unit intervals):

```
intensity_ranges = [
    (0,10), (10,20), ..., (240,255)  # 25 total bins
]
```

   - Quantified dark pixels using thresholds: < 30, < 40, < 50, < 60 intensity
3. **Wing-Wingtip Difference**
   - Computed percentage of wingtip pixels darker than mean wing intensity of the bird.
   - **Thresholds**: Calculated the proportion of wingtip pixels exceeding specific difference thresholds (10, 20, ..., 100) compared to the mean wing intensity, quantifying the contrast between regions.

4.9.2 Statistical Testing Methodology

For quantitative comparison between the two gull species, we employed Welch's t-test (unequal variances t-test) to analyze the differences in wing and wingtip intensity values. This test was selected as it does not assume equal variances between groups, making it more robust for our dataset.

The test statistic was calculated as:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{4.18}$$

where $\bar{X}_i$, s_i^2, and n_i are the means, sample variances, and sizes for Slaty-backed and Glaucous-winged Gulls, respectively. Degrees of freedom (ν) were approximated using the Welch–Satterthwaite equation.

The degrees of freedom were approximated using the Welch–Satterthwaite equation:

$$\nu \approx \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}} \tag{4.19}$$

Where ν represents the degrees of freedom used to determine the critical values for the t-distribution.

All statistical tests were implemented using SciPy's `stats.ttest_ind()` function with `equal_var=False` parameter to specify Welch's t-test. These statistical analyses allowed us to find and objectively validate whether the differences found through the analysis in areas highlighted by Grad-CAM (wing and wingtip) between species are quantitatively significant.

The entire analysis pipeline was implemented in Python, utilizing libraries including OpenCV for image processing, NumPy and Pandas for data manipulation, and SciPy for statistical testing. Various visualization plots were created to represent the analysis results that include threshold-based differences, darkness metrics, pixel intensity distributions. per-image variability, and wing vs wingtip intensity relationships.

4.10 Verification by Clustering for Species Differentiation

To validate and complement the results achieved through the analysis of the region-specific features that were significantly different among the 2 species, a comprehensive clustering analysis framework was implemented that leverages traditional machine learning techniques to check whether the two species can be classified using the features. This approach helps validate the discriminative features identified by the deep learning models that were quantified.

4.10.1 Feature Extraction and Preprocessing

The clustering analysis utilizes three key morphological features extracted from the wingtip regions, specifically **Mean wing intensity**, **Mean wingtip intensity**, and **Per-**

centage of pixels darker than intensity value 60. Although local Binary Pattern (LBP) analysis identified features such as contrast and energy although the difference were not as significant as intensity values. So they were not used in clustering.

Prior to clustering analysis, the following preprocessing steps were implemented:

1. **Data Loading**: The dataset was loaded from the CSV file (produced by intensity analysis script), with features and species labels separated for analysis.
2. **Feature Standardization**: All features were standardized using `StandardScaler` to ensure each feature contributed equally to the clustering algorithms, preventing features with larger magnitudes from dominating the analysis.
3. **Dimensionality Reduction**: Principal Component Analysis (PCA) was applied to reduce the three-dimensional feature space to two dimensions for visualization purposes. The explained variance ratio was calculated to assess how much information was preserved in the lower-dimensional representation.

4.10.2 Clustering Algorithms Implementation

Three distinct clustering algorithms were implemented to provide a comprehensive evaluation of the feature space with Number of clusters/components set to 2 (corresponding to the two gull species).

K-means Clustering

K-means clustering was implemented with multiple initializations (`n_init = 10`) to avoid local optima. K-means partitions the data by minimizing the within-cluster sum of squares. The algorithm iteratively assigns data points to the nearest cluster centroid and then recalculates the centroids based on the new cluster assignments until convergence is achieved.

Hierarchical Clustering

Hierarchical clustering was implemented with Ward's linkage. This approach builds a hierarchy of clusters. Initially, each data point is treated as a single cluster. The algorithm then iteratively merges the closest pairs of clusters until only the specified number of clusters remains. Ward's linkage was chosen to minimize the variance within the newly formed clusters at each merge, promoting the creation of compact, spherical clusters.

Gaussian Mixture Model

GMM assumes that the data is generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The algorithm estimates the parameters of these distributions and assigns data points to clusters based on the probability that they belong to each distribution. This probabilistic approach allows for more flexibility in cluster shapes compared to K-means.

4.10.3 Evaluation Metrics

The effectiveness of each clustering algorithm was evaluated using multiple metrics:

- **Silhouette Score**: Measures how similar a data point is to its own cluster com-

pared to other clusters. Values range from -1 to 1, with higher values indicating better-defined clusters [59].
- **Adjusted Rand Index (ARI)**: Measures the similarity between two data clusterings, adjusting for chance. Values range from -1 to 1, with higher values indicating greater similarity.
- **Clustering Accuracy**: Calculated as the percentage of specimens correctly classified after mapping clusters to the majority species within each cluster.
- **Confusion Matrix**: Visualizes the number of correctly and incorrectly clustered specimens after mapping cluster labels to species labels.

4.10.4 Cluster Mapping and Misclassification Analysis

To interpret the clustering results in terms of species classification:

1. **Cluster-to-Species Mapping**: For each cluster, the majority species was identified, creating a mapping from cluster labels to species labels.
2. **Misclassification Analysis**: Using this mapping, specimens were identified as correctly or incorrectly clustered. Misclassified specimens were visualized in the PCA space and exported to CSV files for further analysis.

4.10.5 Visualization and Interpretation

Multiple visualizations were generated to aid interpretation:

1. **PCA Plots**: Showing the distribution of specimens in the reduced two-dimensional space with color-coding by cluster assignment.
2. **Cluster Centers**: For K-means and GMM, cluster centers were plotted in the PCA space to visualize the centroids of each cluster.
3. **Misclassification Visualization**: Highlighting correctly and incorrectly clustered specimens in the PCA space to identify areas of confusion.
4. **Feature Importance Plots**: Bar charts showing the relative importance of each feature based on cluster center differences (for K-means).
5. **Algorithm Comparison**: Bar charts comparing the silhouette scores across all clustering algorithms to assess their performance.

This comprehensive clustering analysis provides valuable insights into the natural grouping of morphological features and helps validate the discriminative power of the significantly different features found from the analysis performed on regions of the birds highlighted by Grad-CAM.

4.10.6 Implementation Details

The entire analytical workflow was implemented in Python using the following libraries:

- `scikit-learn` for clustering algorithms, PCA, and evaluation metrics
- `pandas` for data management
- `NumPy` for numerical operations
- `Matplotlib` and `Seaborn` for visualization

A consistent random state (42) was used throughout the analysis to ensure reproducibility. Results, including visualizations and misclassification data, were automatically saved to an output directory for documentation and further analysis.

This comprehensive methodology allowed for a robust evaluation of whether the extracted statistical features could effectively distinguish between the two gull species through unsupervised clustering, providing valuable insights into feature significance for species classification.

5 Results and Discussion

5.1 Model Performance of top-performing models trained on latest Stage 3 dataset

Table 5.1: Validation and Test Accuracy Comparison

Model	Val. Acc. (%)	Test Acc. (%)
VGG-16	97.37	95.74
ViT	96.04	93.62
Inception v3	97.80	91.49
ResNet50	96.04	82.98
EnhancedViT	94.52	95.74
InterpretableViT	94.74	95.74

Table 5.2: Comprehensive Model Performance Metrics on Test Dataset

Model	Glaucous Winged			Slaty Backed		
	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)
VGG-16	87.50	100.00	93.33	100.00	93.94	96.88
ViT	100.00	78.57	88.00	91.67	100.00	95.65
Inception v3	77.78	100.00	87.50	100.00	87.88	93.55
ResNet50	66.67	85.71	75.00	93.10	81.82	87.10
EnhancedViT	87.50	100.00	93.33	100.00	93.94	96.88
InterpretableViT	92.86	92.86	92.86	96.97	96.97	96.97

Based on the results presented in Tables 5.1 and 5.2, it is evident that several models achieved high validation and test accuracy although there were notable differences in their interpretability and localization capabilities. The results demonstrate that deep learning models, particularly VGG-16, achieved high classification accuracy in distinguishing Slaty-backed and Glaucous-winged Gulls, validating the efficacy of transfer learning for fine-grained avian classification.

However, qualitative analysis of Grad-CAM and interpretability visualizations Section 8 revealed that most models, with the exception of VGG-16, failed to consistently highlight the biologically relevant regions-namely, the wings and wingtip areas-of the gull images. Therefore, VGG-16 was selected as the best-performing model.

The Grad-CAM visualizations of VGG-16 model as shown in Figure 5.1, confirmed that models consistently focused on wing and wingtip regions, mirroring ornithological expertise. For instance, in Slaty-backed Gulls, high activation coincided with dark primary feather patterns, while Glaucous-winged Gulls showed activations around lighter wing and wingtips. This congruence between model attention and expert knowledge not only validates the models' decision-making but also reinforces the biological relevance of wingtip and wing morphology as a taxonomic marker aligning with the biolog-

ical reality that diagnostic traits in these species are localized to specific morphological regions rather than global image contexts.

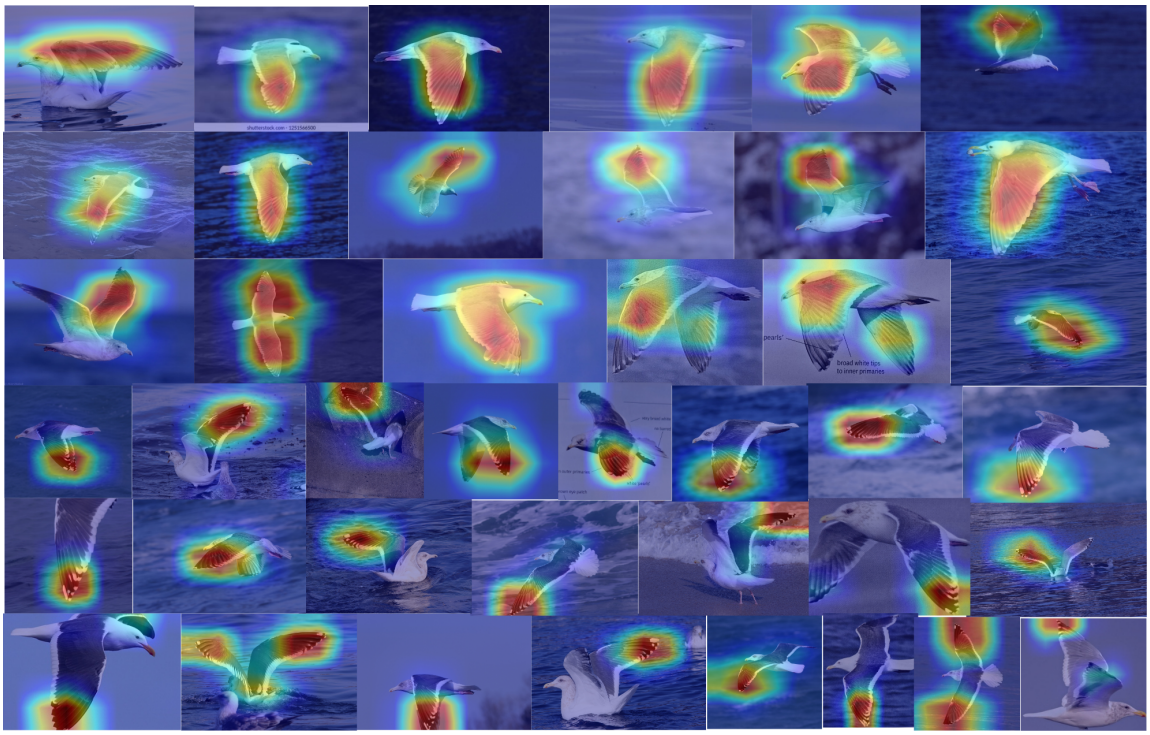


Figure 5.1: Grad-CAM visualizations for a selection of unseen test images highlighting wing and wingtip regions of the VGG-16 model

5.2 Intensity Analysis Results

5.2.1 Wing and Wingtip Intensity Analysis

The intensity analysis revealed significant quantifiable differences in wing and wingtip intensities between Slaty-backed Gulls and Glaucous-winged Gulls, providing strong discriminative features for species identification.

Statistical analysis of both wing and wingtip intensity demonstrated clear and significant differences between the two gull species across multiple samples. Independent samples t-tests (Welch's t-tests, allowing for unequal variances) were performed for both wing and wingtip intensities, comparing the means of the two species. Both tests confirmed that this difference is highly significant ($p < 0.001$), indicating that the observed differences in intensity for both wing and wingtip are extremely unlikely to have arisen by chance.

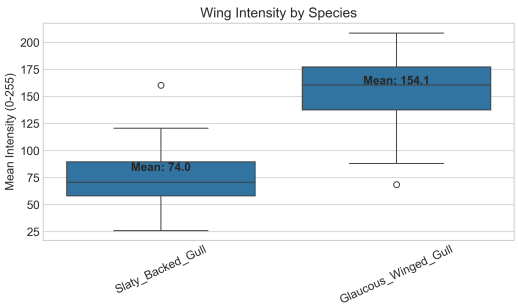
Table 5.3: Wing and Wingtip Intensity Statistical Summary

Feature	Species	Mean Intensity	Std. Dev.	% Difference	Statistical Significance
Wing	Slaty-backed Gull Glaucous-winged Gull	73.98 154.10	21.90 30.82	108.3%	$p < 0.001$ (t-test)
Wingtip	Slaty-backed Gull Glaucous-winged Gull	81.56 155.36	20.66 32.52	90.5%	$p < 0.001$ (t-test)

Wing Intensity

Figures 5.2a and 5.2b show that Glaucous-winged Gulls have consistently higher and less variable wing intensity compared to Slaty-backed Gulls. The distributions are largely non-overlapping, making mean wing intensity a robust distinguishing feature between the species.

(a) Comparison of wing intensity values between species.

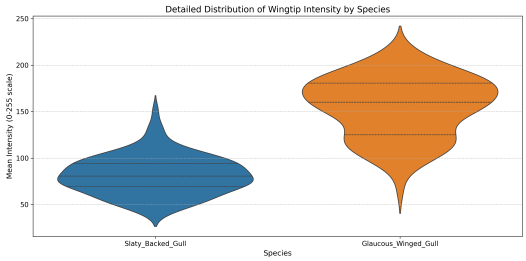


(b) Mean wing intensity across samples.

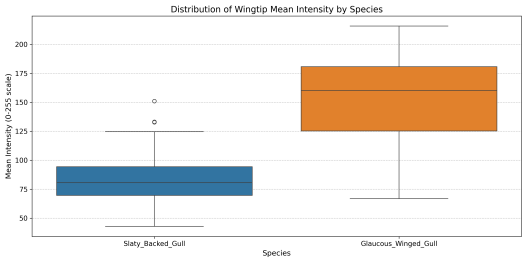
Figure 5.2: Wing intensity metrics for Slaty-backed and Glaucous-winged Gulls, showing significant brightness differences.

Wingtip Intensity

Figures 5.3a and 5.3b illustrate that, while wingtip intensity also differs significantly between species, there is greater variability and overlap, as indicated by the broader range and outliers. This variability is likely due to the presence of white spots and differences in darkness, making wingtip intensity less consistently robust than wing intensity for species differentiation.



(a) Wingtip intensity violin plot.



(b) Wingtip intensity distribution.

Figure 5.3: Wingtip intensity metrics for Slaty-backed and Glaucous-winged Gulls, highlighting greater variability and overlap between species.

5.2.2 Distributions of Wing and Wingtip Intensities

A comprehensive analysis of pixel intensity distributions in both the wing and wingtip regions reveals clear, species-specific patterns that distinguish Slaty-backed Gulls from Glaucous-winged Gulls. This section presents and interprets the observed distributions using multiple complementary metrics.

5.2.3 Wing Intensity Distributions

Figure 5.4 displays the distribution of mean wing intensities for both species. Slaty-backed Gulls cluster at lower mean intensity values, indicating darker wings, while Glaucous-winged Gulls are concentrated at higher mean intensities, reflecting lighter wings. The distributions are largely non-overlapping, underscoring a strong differentiation in overall wing brightness between the two species.

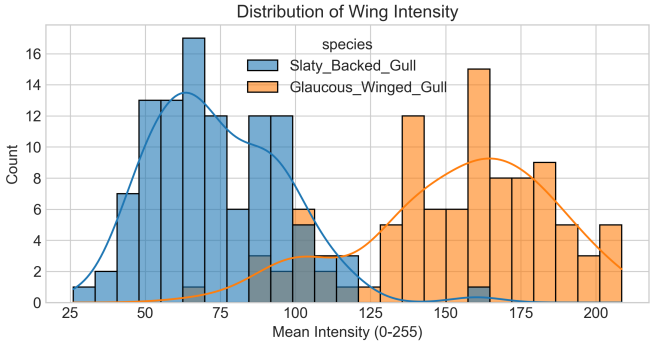
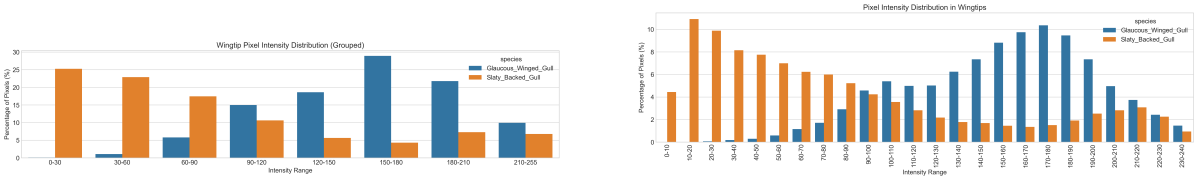


Figure 5.4: Distribution of mean wing intensity for Slaty-backed Gulls (blue) and Glaucous-winged Gulls (orange). Each species forms a distinct cluster, reflecting their characteristic wing brightness.

5.2.4 Wingtip Intensity Distributions

The distributions of wingtip pixel intensities for both species are shown in Figure 5.5. Slaty-backed Gulls (orange) have a higher proportion of darker pixels (lower intensity values), while Glaucous-winged Gulls (blue) show a greater proportion of lighter pixels (higher intensity values). Both grouped and fine-grained binning approaches reveal unique, largely non-overlapping patterns for each species.



(a) Grouped intensity bins

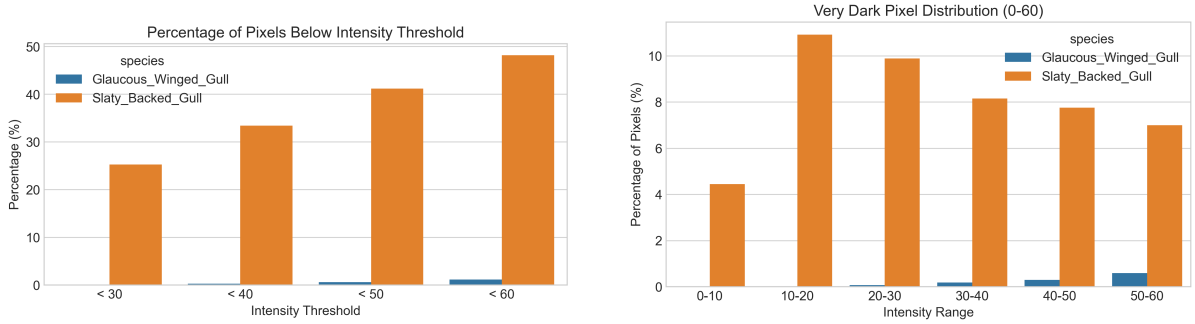
(b) Fine-grained intensity bins

Figure 5.5: Distributions of wingtip pixel intensities for Slaty-backed Gulls (orange) and Glaucous-winged Gulls (blue) using (a) grouped and (b) fine-grained intensity bins.

5.2.5 Presence of Very Dark Pixels

A detailed analysis of wingtip pixel intensity reveals a striking difference in the prevalence of very dark pixels between Slaty-backed Gulls and Glaucous-winged Gulls. As shown in Figure 5.6a, Slaty-backed Gulls have a much higher percentage of wingtip pixels below each intensity threshold examined (<30, <40, <50, <60), with values rising from 25% to nearly 48%. In contrast, Glaucous-winged Gulls have negligible proportions of such dark pixels at all thresholds.

Figure 5.6b further demonstrates that this difference is consistent across all sub-ranges within the very dark pixel spectrum (0–60), with Slaty-backed Gulls consistently showing a substantially greater percentage of pixels in each bin.



(a) Percentage of wingtip pixels below each intensity threshold.

(b) Distribution of very dark pixels (0–60) in wingtip regions.

Figure 5.6: Comparison of very dark pixel proportions in wingtip regions for Slaty-backed and Glaucous-winged Gulls.

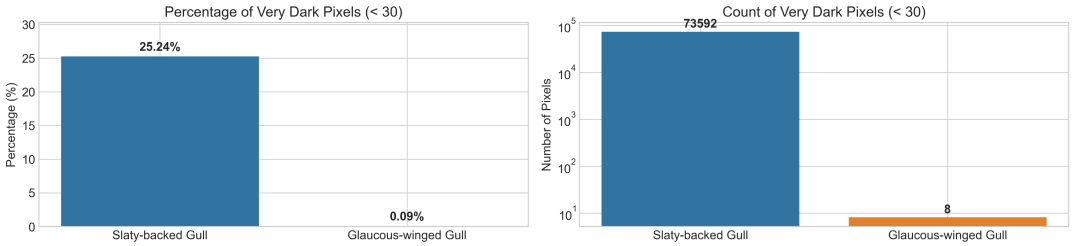


Figure 5.7: Left: Percentage of very dark pixels ($<$30) by species. Right: Average count of very dark pixels ($<$30) per wingtip region, shown on a logarithmic scale.

The quantitative difference in very dark pixels between species is substantial, as shown in Figure 5.7, with Slaty-backed Gulls having on average 73,592 very dark pixels compared to just 8 in Glaucous-winged Gulls. This represents a critical diagnostic feature for species identification. This pronounced disparity represents a critical diagnostic feature for species identification. However, it is important to note that the presence of very dark pixels is not consistent across all images, so this feature may not be used as a distinguishing criterion in every case.

5.2.6 Relationship Between Wing and Wingtip Intensities

To quantitatively assess the relative darkness of wingtip area compared to the rest of the wing which were the 2 critical areas highlighted by Grad-CAM, the proportion of wingtip pixels that are darker than the mean wing intensity for each species was calculated as suggested by industry expert Professor Gibbins. The results are presented in Table 5.4.

Table 5.4: Proportion of Wingtip Pixels Darker than Mean Wing Intensity

Species	Percentage
Slaty-backed Gull	56.69%
Glaucous-winged Gull	47.71%

Figure 5.8 visualizes the relationship between mean wing intensity and mean wingtip intensity for individual birds. The dashed line indicates equal intensity between wing and wingtip. As illustrated, there is a strong positive correlation between wing and wingtip intensities across both species; as the mean wing intensity increases, the mean wingtip intensity also increases. This pattern suggests that, in most cases, the relative difference in darkness between the wing and wingtip results only in a slight difference in the proportion of darker wingtip pixels between the two species.

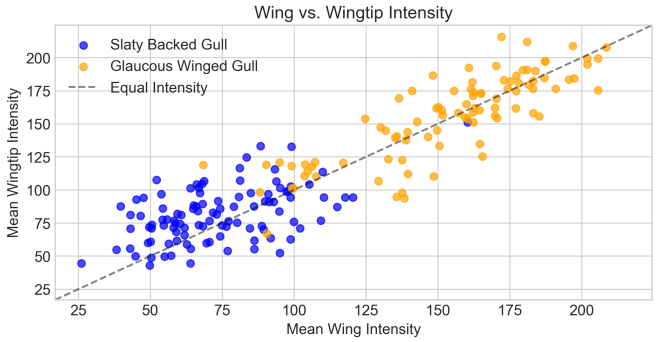
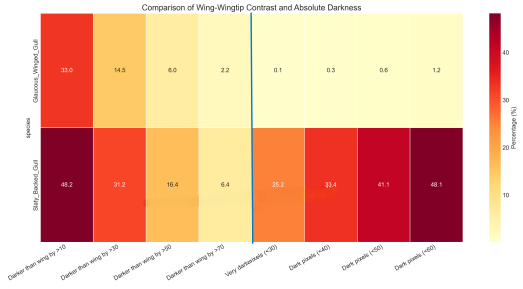


Figure 5.8: Scatter plot of mean wing intensity vs. mean wingtip intensity for Slaty-backed Gulls (blue) and Glaucous-winged Gulls (orange). The dashed line indicates equal intensity between wing and wingtip.

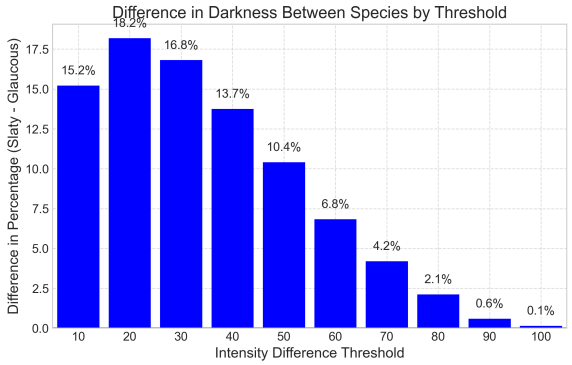
5.2.7 Wingtip Contrast and Darkness Feature Comparison

To comprehensively compare the wingtip characteristics distinguishing Slaty-backed and Glaucous-winged Gulls, we analyzed two key features: (1) the proportion of wingtip pixels that are darker than the wing by varying intensity thresholds, and (2) the proportion of pixels that are absolutely dark, below fixed intensity values.

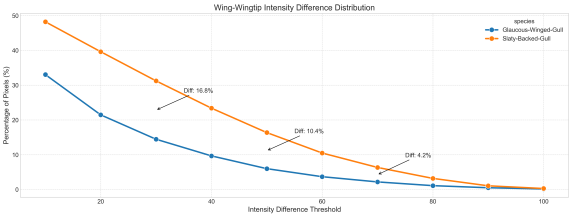
Figure 5.9 presents a set of visualizations summarizing these metrics for both species. The heatmap in subfigure 5.9a displays, on the left, the percentage of wingtip pixels darker than the wing by various thresholds, and on the right, the percentage of absolutely dark pixels. Subfigure 5.9b shows the absolute difference in the proportion of dark pixels between species at each threshold, while subfigure 5.9c plots the difference curve (Slaty-backed minus Glaucous-winged) across thresholds, highlighting where the species gap is largest. Subfigure 5.9d quantifies the relative difference by showing the ratio of dark pixel proportions (Slaty-backed/Glaucous-winged) at each threshold, with the ratio peaking near 2.8–2.9$\times$ between thresholds of 50 and 80. Collectively, these visualizations demonstrate that Slaty-backed Gulls consistently have a much higher proportion of both strongly contrasting and absolutely dark pixels, making these features highly discriminative.



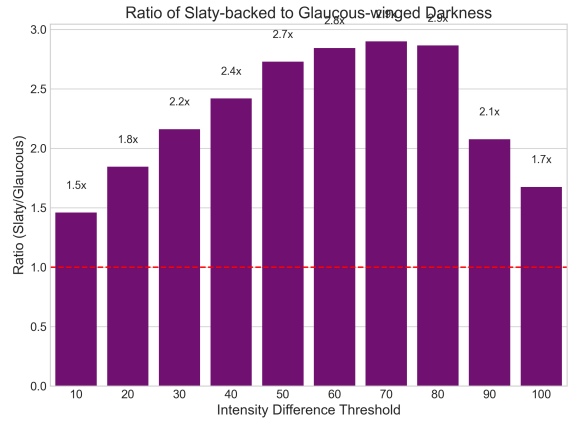
(a) Heatmap of wingtip darkness and contrast features for each species.



(b) Absolute difference in percentage of dark pixels by threshold.



(c) Difference curve (Slaty-backed minus Glaucous-winged) across thresholds.



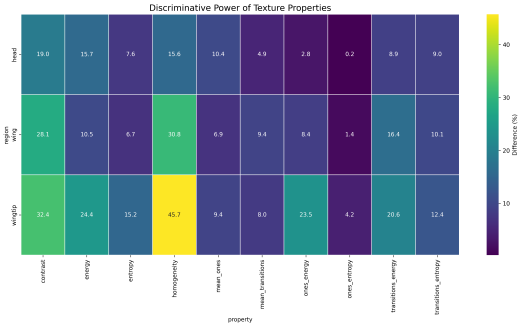
(d) Ratio of dark pixel proportions (Slaty-backed/Glaucous-winged) by threshold.

Figure 5.9: Summary of wingtip darkness and contrast metrics distinguishing Slaty-backed and Glaucous-winged Gulls. (a) Heatmap of key features; (b) absolute difference by threshold; (c) difference curve; (d) ratio of proportions. These visualizations collectively highlight the strong and persistent contrast in wingtip darkness between the two species.

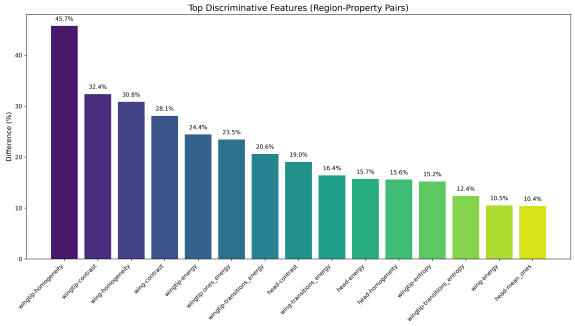
5.3 Results: Local Binary Pattern Texture Analysis

For the final results displayed below, the default LBP variant was selected since default histogram retains all LBP codes, providing a comprehensive representation of local texture patterns. Consequently, it facilitates the direct calculation of derived features such as the number of ones and the number of transitions per pattern, as these statistics can be computed from the complete set of binary codes present in the default histogram. This approach ensures that no potentially discriminative texture information is excluded, thereby supporting a more detailed and flexible analysis of texture complexity and structure across the studied regions.

5.3.1 Discriminative Power Analysis



(a) Regional discriminative power of texture properties



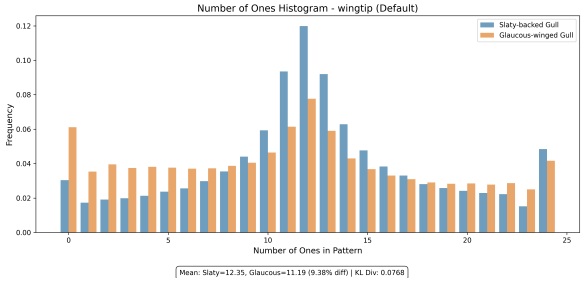
(b) Top discriminative features.

Figure 5.10: Discriminative power analysis of LBP texture features: (a) heatmap of regional differences, (b) top discriminative features.

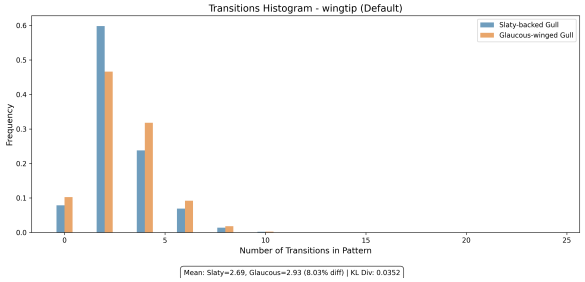
The discriminative power of LBP texture features is visualized as a heatmap and bar chart in Figure 5.10, highlighting wingtip homogeneity (45.74%), wingtip contrast (32.36%), and wing homogeneity (30.83%) as the top three discriminative features.

5.3.2 Limitations of standard LBP code analysis

Although standard texture properties demonstrated substantial differences between species (with top feature close to 50% difference), it is important to note that the features from LBP pattern codes are not guaranteed to be rotationally invariant. The abstract features (Number of Ones and Transitions) showed more modest discriminative power (maximum 10.4%) but can be considered rotation variant since we are only dealing with the presence or count of ones or transitions in a region.

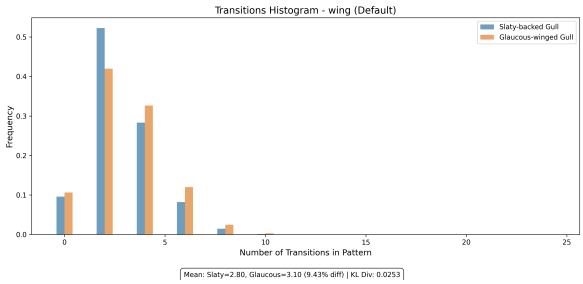


(a) Wingtip Number of Ones distribution (9.38% difference)

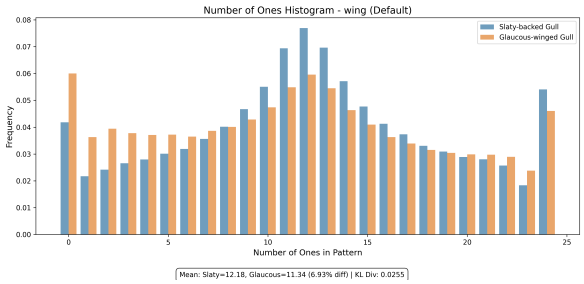


(b) Wingtip Transitions distribution (8.0% difference)

Figure 5.11: Histograms of LBP features for the wingtip region: (a) Number of Ones, (b) Transitions.



(a) Wing Transitions distribution (9.4% difference)



(b) Wing Number of Ones distribution (7.5% difference)

Figure 5.12: Histograms of LBP features for the wing region: (a) Transitions, (b) Number of Ones.

The distributions of the 'Number of Ones' and 'Transitions' features for the wingtip and wing regions are shown as histograms in Figures 5.11 and 5.12, respectively.

5.3.3 Abstract Feature Distributions

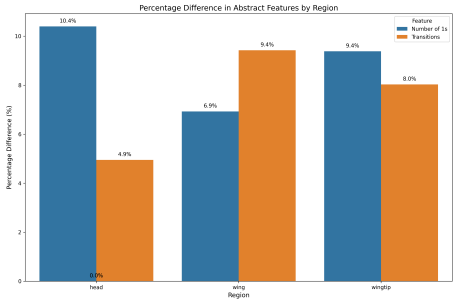


Figure 5.13: Abstract feature differences, showing head region's Number of Ones as most discriminative (10.4% difference).

The figures illustrate the results of abstract features across different regions. Figure 5.13 presents a bar chart highlighting the most discriminative feature from 1s and transitions histogram, showcasing their respective differences.

5.3.4 Top Discriminative Features from 1s and Transitions Histograms

Table 5.5: Top discriminative region-histogram features ranked by percentage difference.

Rank	Region	Histogram Feature	Statistic	Difference (%)
1	Wingtip	Ones	Energy	23.5
2	Wingtip	Transitions	Energy	20.6
3	Wing	Transitions	Energy	16.4

To further quantify these observations and analyse the texture features calculated from the histograms, Table 5.5 lists the top discriminative features based on their percentage differences between species. The analysis demonstrates that LBP texture features, particularly in the wingtip and wing region, can provide robust discriminative power for gull species identification although the difference is very small.

5.4 Clustering Results

5.4.1 Clustering Performance Metrics

The performance of each clustering algorithm was evaluated using both unsupervised and supervised metrics. Table 5.6 presents a comprehensive comparison of the three clustering approaches.

Table 5.6: Performance and confusion matrix summary for clustering algorithms. TP = True Positives, FN = False Negatives, GWG = Glaucous-Winged Gull, SBG = Slaty-Backed Gull.

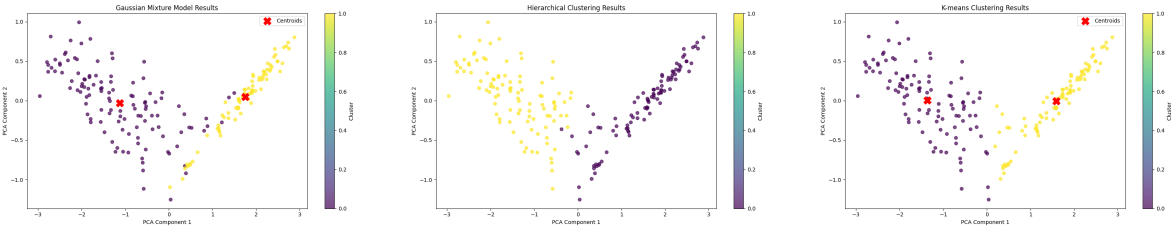
Algorithm	Silhouette	ARI	Accuracy	TP (GWG)	FN (GWG)	TP (SBG)	FN (SBG)
K-means	0.605	0.843	95.94%	88	5	101	3
Hierarchical	0.594	0.881	96.95%	92	1	99	5
GMM	0.548	0.717	92.39%	78	15	104	0

Table 5.6 summarizes the key performance metrics and confusion matrix results for all three clustering algorithms. The silhouette score (ranging from -1 to 1) measures how well-separated the clusters are, with higher values indicating better-defined clusters. The Adjusted Rand Index (ARI) measures the similarity between the clustering results and the ground truth labels, with values closer to 1 indicating better alignment with the actual species classification.

The prominence of wing intensity features aligns with ornithological knowledge that Slaty-Backed Gulls typically exhibit darker wing patterns compared to Glaucous-Winged Gulls, providing a data-driven validation of these distinguishing characteristics.

5.4.2 Integration with Visual Analysis

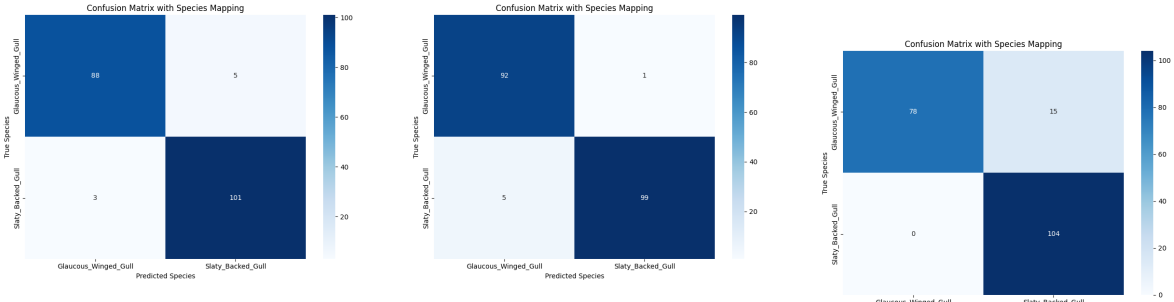
The clustering performance is visualized through two complementary perspectives: the cluster formations in reduced PCA space and the classification accuracy via confusion matrices considering majorily class to be the true class of the cluster. Figure 5.14 demonstrates how each clustering algorithm partitions the data in the PCA-reduced feature space. The K-means visualization (Figure 5.14c) clearly shows two well-defined clusters with distinct centroids, while hierarchical clustering (Figure 5.14b) exhibits similar separation but with slightly different cluster boundaries. The GMM results (Figure 5.14a) show more density-based partitioning that accommodates the natural distribution of the data points.



(a) GMM Clustering (b) Hierarchical (c) K-means

Figure 5.14: Cluster formations in PCA-reduced space showing the distribution of data points with centroids marked for K-means and GMM. All three algorithms show clear separation between the two species along PCA Component 1.

5.4.3 Confusion Matrix Analysis



(a) K-means (Accuracy: 95.94%) (b) Hierarchical (Accuracy: 96.95%) (c) GMM (Accuracy: 92.39%)

Figure 5.15: Comparative confusion matrices showing classification performance across clustering algorithms, with hierarchical clustering achieving the highest overall accuracy.

The confusion matrices in Figure 5.15 provide detailed insights into each algorithm's classification performance. The hierarchical clustering matrix (Figure 5.15b) shows excellent classification of Glaucous-Winged Gulls with only 1 false negative, while K-means (Figure 5.15a) shows balanced performance across both species. The GMM confusion matrix (Figure 5.15c) reveals perfect recall for Slaty-Backed Gulls but struggles more with correctly classifying Glaucous-Winged Gulls.

Notably, all three algorithms achieved over 92% accuracy, with hierarchical clustering performing best at nearly 97%. The high clustering accuracy achieved demonstrates that the three extracted morphological features (mean wing intensity, mean wingtip intensity, and percentage of pixels darker than intensity value 60) provide strong discriminative power for distinguishing between Slaty-Backed Gulls and Glaucous-Winged Gulls. This confirms that the features identified through the CNN-based analysis and subsequent statistical extraction are biologically meaningful and can effectively capture species-specific characteristics. The fact that unsupervised clustering can effectively discriminate between these visually similar gull species based solely on extracted image features proves that the intensity features analysed are significant and can be used for classification thereby validating the statistical features.

6 Conclusion

The dissertation has successfully addressed its primary research aims. First, high-accuracy classifiers, particularly VGG-16 achieving a test accuracy of 95.74%, were developed using transfer learning, demonstrating that pre-trained convolutional neural network architectures can excel in fine-grained gull classification even with limited data. Iterative dataset refinement from expert-curated images was vital, minimizing noise from age-related and environmental variability. Second, interpretability was achieved through Grad-CAM visualizations, which effectively bridged artificial intelligence and biological expertise by highlighting the wing and wingtip regions. This transparency is essential for building trust in automated taxonomic tools. Third, statistical and clustering analyses confirmed significant and quantifiable differences in wing and wingtip intensity metrics, empirically supporting field identification criteria while also revealing nuances such as interspecific overlap in wingtip intensity that require expert contextualization. Finally, the dominance of wingtip features in both model attention maps and statistical analysis suggests that artificial intelligence can not only replicate human expertise but also uncover underappreciated diagnostic traits, such as the prevalence of very dark pixels in Slaty-backed Gulls. Collectively, these achievements demonstrate the potential of integrating deep learning with interpretability and quantitative analysis to advance fine-grained species identification and provide new insights into avian taxonomy.

Future research can expand to hybrid specimens and juvenile plumage while testing robustness across diverse imaging conditions or performing other analyses to find more significantly different features. By integrating computational power with ecological knowledge, this approach offers a paradigm for addressing the taxonomic impediment in rapidly evolving species complexes.

7 Bibliography

[1] G. Ceballos, P. R. Ehrlich, A. D. Barnosky, A. García, R. M. Pringle, and T. M. Palmer, "Biological annihilation via the ongoing sixth mass extinction signaled by vertebrate population losses and declines," *Proceedings of the National Academy of Sciences*, vol. 114, no. 30, pp. E6089–E6096, 2017. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1704949114

[2] C. Coleman, "Taxonomy in times of the taxonomic impediment - examples from the community of experts on amphipod crustaceans," *Journal of Crustacean Biology*, vol. 35, pp. 729–740, 11 2015.

[3] K. Wang, F. Yang, Z. Chen, Y. Chen, and Y. Zhang, "A fine-grained bird classification method based on attention and decoupled knowledge distillation," *Animals*, vol. 13, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2076-2615/13/2/264

[4] P. Adriaens, M. Muusse, P. J. Dubois, and F. Jiguet, *Gulls of Europe, North Africa, and the Middle East: An Identification Guide*. Princeton and Oxford: Princeton University Press, 2022.

[5] A. Ayyash, *The Gull Guide*. Princeton University Press, 2024.

[6] M. Valan, "Automated image-based taxon identification using deep learning," *Journal of Taxonomy Research*, vol. 45, pp. 123–135, 2023.

[7] W. Lu, Y. Yang, and L. Yang, "Fine-grained image classification method based on hybrid attention module," *Frontiers in Neurorobotics*, vol. 18, p. 1391791, 2024.

[8] R. C. W. M. Muazin Hilal Hasibuan, Novanto Yudistira, "Large-scale bird species classification using cnns," *Nature Machine Intelligence*, vol. 5, pp. 89–101, 2022.

[9] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, 2021.

[10] M. F. Santiago Martinez, "Comparative analysis of deep learning architectures for fine- grained bird classification," July 2024. [Online]. Available: http://essay.utwente.nl/101313/

[11] M. Alswaitti, L. Zihao, W. Alomoush, A. Alrosan, and K. Alissa, "Effective classification of birds' species based on transfer learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 4, pp. 4172–4184, 2025.

[12] A. Alfatemi, S. A. Jamal, N. Paykari, M. Rahouti, and A. Chehri, "Multi-label classification with deep learning and manual data collection for identifying similar bird species," *Procedia Computer Science*, vol. 246, pp. 558–565, 2024, 28th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES 2024). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092402502X

[13] A. Kumar and S. D. Das, “Bird species classification using transfer learning with multistage training,” in *Computer Vision Applications*, C. Arora and K. Mitra, Eds. Singapore: Springer Singapore, 2019, pp. 28–38.

[14] J. S. B. Pralhad Gavali, “Deep convolutional neural networks for bird species classification,” *Ecological Informatics*, vol. 18, pp. 200–215, 2023.

[15] M. Iman, K. Rasheed, and H. R. Arabnia, “A review of deep transfer learning and recent advancements,” *arXiv preprint arXiv:2201.09679*, 2022. [Online]. Available: https://arxiv.org/abs/2201.09679

[16] H.-T. Vo, N. N. Thien, and K. C. Mui, “Bird detection and species classification: Using yolov5 and deep transfer learning models,” *International Journal of Advanced Computer Science and Applications*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260674777

[17] L. I. Mochurad and S. Svystovych, “A new efficient classifier for bird classification based on transfer learning,” *Journal of Engineering*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267561131

[18] T. M. Mohamed and I. M. Alharbi, “Efficient drones-birds classification using transfer learning,” *2023 4th International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, pp. 15–19, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:264976761

[19] G. Ren, “Monkeypox disease detection with pretrained deep learning models,” *Inf. Technol. Control.*, vol. 52, pp. 288–296, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259931761

[20] M. M. Rahman, A. A. Biswas, A. Rajbongshi, and A. Majumder, “Recognition of local birds of bangladesh using mobilenet and inception-v3,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 8, 2020. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2020.0110840

[21] A. A. Biswas, M. M. Rahman, A. Rajbongshi, and A. Majumder, “Recognition of local birds using different cnn architectures with transfer learning,” in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, 2021, pp. 1–6.

[22] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[23] M. M. H. Hasibuan, N. Yudistira, and R. C. Wihandika, “Large scale bird species classification using convolutional neural network with sparse regularization,” in *Proceedings of the 2022 Brawijaya International Conference (BIC 2022)*, ser. Advances in Economics, Business and Management Research, Y. A. Yusran *et al.*, Eds., vol. 235. Atlantis Press, 2023, pp. 651–663. [Online]. Available: https://www.atlantis-press.com/article/125986223.pdf

[24] T. Carneiro, R. V. M. Da Nobrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance analysis of Google Colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, vol. 6, pp. 61 677–61 685, 2018.

[25] A. Karpathy, "A recipe for training neural networks," http://karpathy.github.io/2019/04/25/recipe/, 2019, accessed: 2025-04-28.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Lake Tahoe, Nevada: Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[27] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017. [Online]. Available: https://arxiv.org/abs/1712.04621

[28] Y. Wu and K. He, "Group normalization," 2018. [Online]. Available: https://arxiv.org/abs/1803.08494

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[30] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016. [Online]. Available: https://arxiv.org/abs/1609.04836

[31] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018. [Online]. Available: https://arxiv.org/abs/1804.07612

[32] L. Prechelt, *Early Stopping - But When?* Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 55–69. [Online]. Available: https://doi.org/10.1007/3-540-49430-8_3

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[34] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, 1992, pp. 950–957. [Online]. Available: https://papers.nips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf

[35] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0192-5

[36] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608018302107

[37] S. Yang *et al.*, "Image data augmentation for deep learning: A survey," *arXiv preprint arXiv:2204.08610*, 2022. [Online]. Available: https://arxiv.org/abs/2204.08610

[38] Z. Wu, "Image data augmentation techniques based on deep learning: A survey," *Mathematical Biosciences and Engineering*, vol. 21, no. 6, pp. 6190–6224, 2024.

[39] Y. Cui, M. Jia, T.-Y. Lin, and Y. Song, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9268–9277. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/papers/Cui_Class-Balanced_Loss_Based_on_Effective_Number_of_Samples_CVPR_2019_paper.pdf

[40] G. Chu, B. Potetz, W. Wang, A. Howard, Y. Song, F. Brucher, T. Leung, and H. Adam, "Fine-grained bird species recognition using high resolution dcnns," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 281–290.

[41] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[42] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *Artificial Neural Networks and Machine Learning–ICANN 2018*, pp. 270–279, 2018.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[44] A. S. Pearline, V. S. Kumar, S. Harini, S. M. Thampi, and E.-S. M. El-Alfy, "A study on plant recognition using conventional image processing and deep learning approaches," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 3, pp. 1997–2004, 2019.

[45] K. S. M.E., A. D. K. M.E., S. P, and S. Senthilvelan, "Birds species identification using deep learning model," in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024, pp. 1–7.

[46] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[47] R. Wightman, "PyTorch Image Models," https://github.com/rwightman/pytorch-image-models, 2021.

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” 2015. [Online]. Available: https://arxiv.org/abs/1512.00567

[49] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[50] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 7132–7141. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.pdf

[51] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*. PMLR, 2015, pp. 448–456. [Online]. Available: https://arxiv.org/abs/1502.03167

[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” 2015. [Online]. Available: https://arxiv.org/abs/1512.04150

[53] S. Abnar and W. Zuidema, “Quantifying attention flow in transformers,” *arXiv preprint arXiv:2005.00928*, 2020. [Online]. Available: https://arxiv.org/abs/2005.00928

[54] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” *arXiv preprint arXiv:1704.02685*, 2017. [Online]. Available: https://arxiv.org/abs/1704.02685

[55] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, “Sanity checks for saliency maps,” *arXiv preprint arXiv:1810.03292*, Oct 2018. [Online]. Available: https://arxiv.org/pdf/1810.03292.pdf

[56] H. Chefer, S. Gur, and L. Wolf, “Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers,” *arXiv preprint arXiv:2012.09838*, 2021. [Online]. Available: https://arxiv.org/abs/2012.09838

[57] N. Monzón, “Image normalization,” http://dev.ipol.im/~nmonzon/Normalization.pdf, 2017, accessed: 2025-04-29.

[58] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

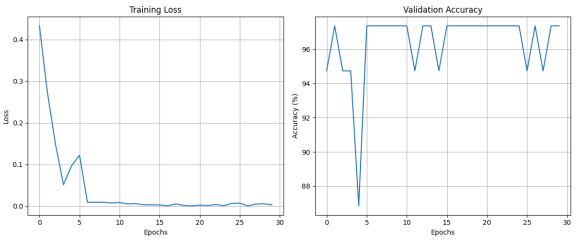
[59] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257

8 Appendix

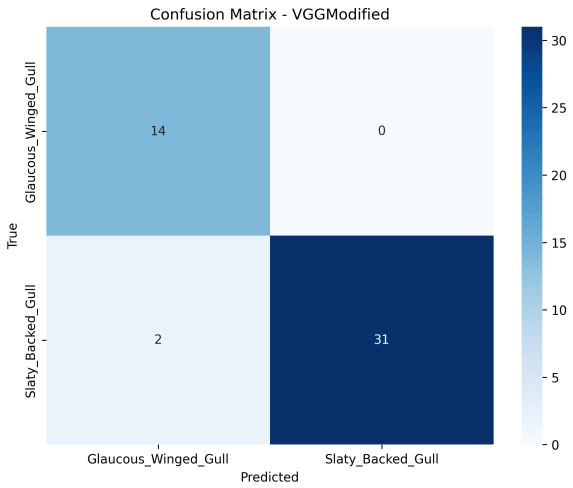
8.1 Model Training Performance

This section presents the training performance graphs and confusion matrices for all models evaluated in this study.

8.1.1 VGG-16



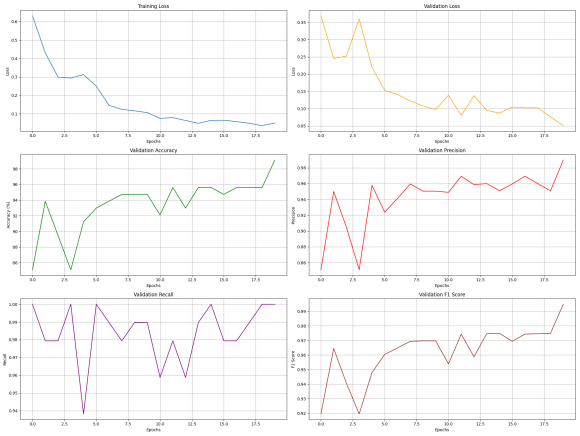
(a) Training and validation loss across epochs



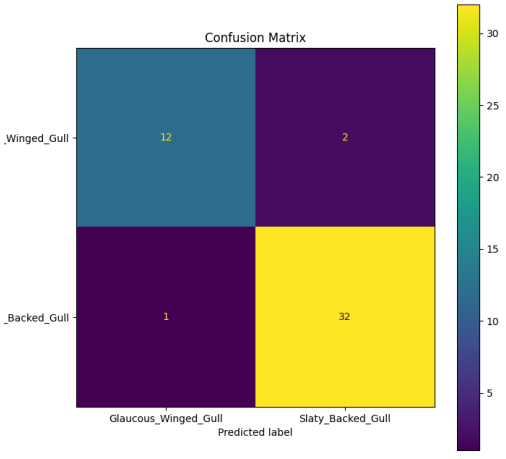
(b) Confusion matrix on test set

Figure 8.1: VGG-16 model performance visualization

8.1.2 Vision Transformer (ViT)



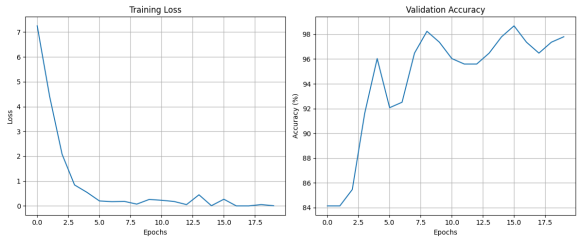
(a) Training and validation loss across epochs



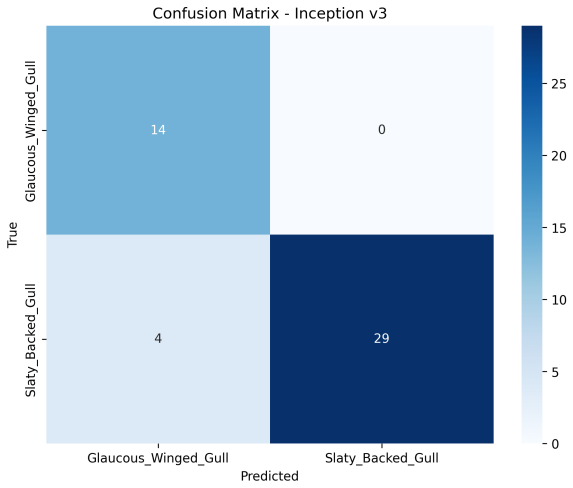
(b) Confusion matrix on test set

Figure 8.2: Vision Transformer model performance visualization

8.1.3 Inception v3



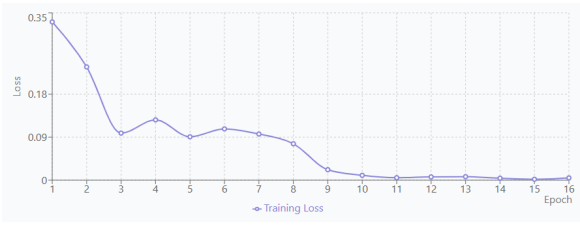
(a) Training and validation loss across epochs



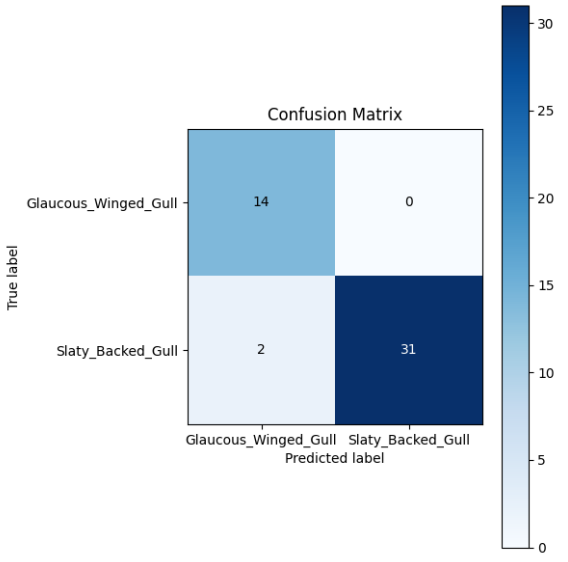
(b) Confusion matrix on test set

Figure 8.3: Inception v3 model performance visualization

8.1.4 Enhanced Vision Transformer



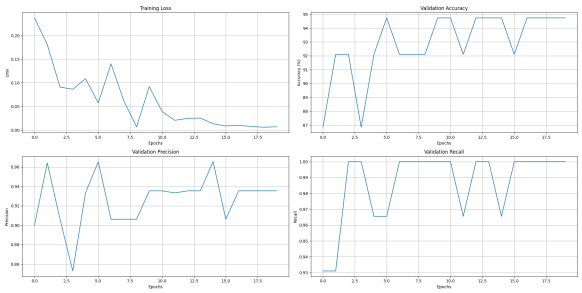
(a) Training and validation loss across epochs



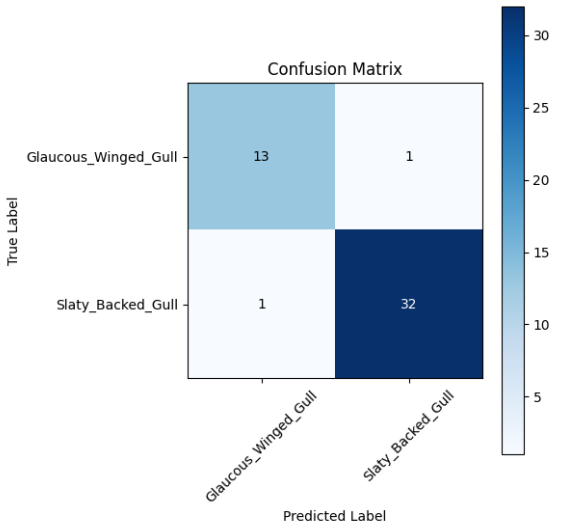
(b) Confusion matrix on test set

Figure 8.4: Enhanced Vision Transformer model performance visualization

8.1.5 Interpretable Vision Transformer



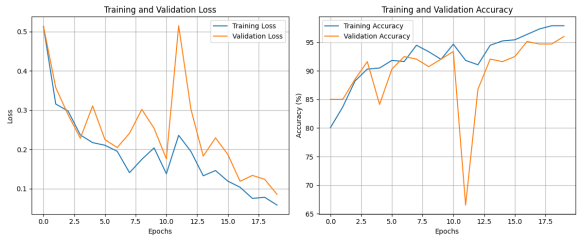
(a) Training and validation loss across epochs



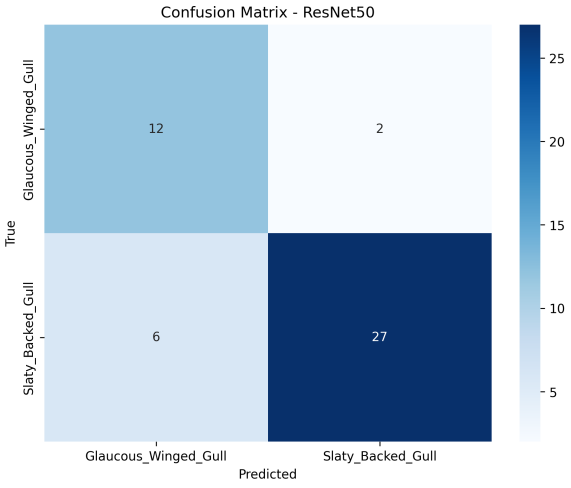
(b) Confusion matrix on test set

Figure 8.5: Interpretable Vision Transformer model performance visualization

8.1.6 ResNet50



(a) Training and validation loss across epochs



(b) Confusion matrix on test set

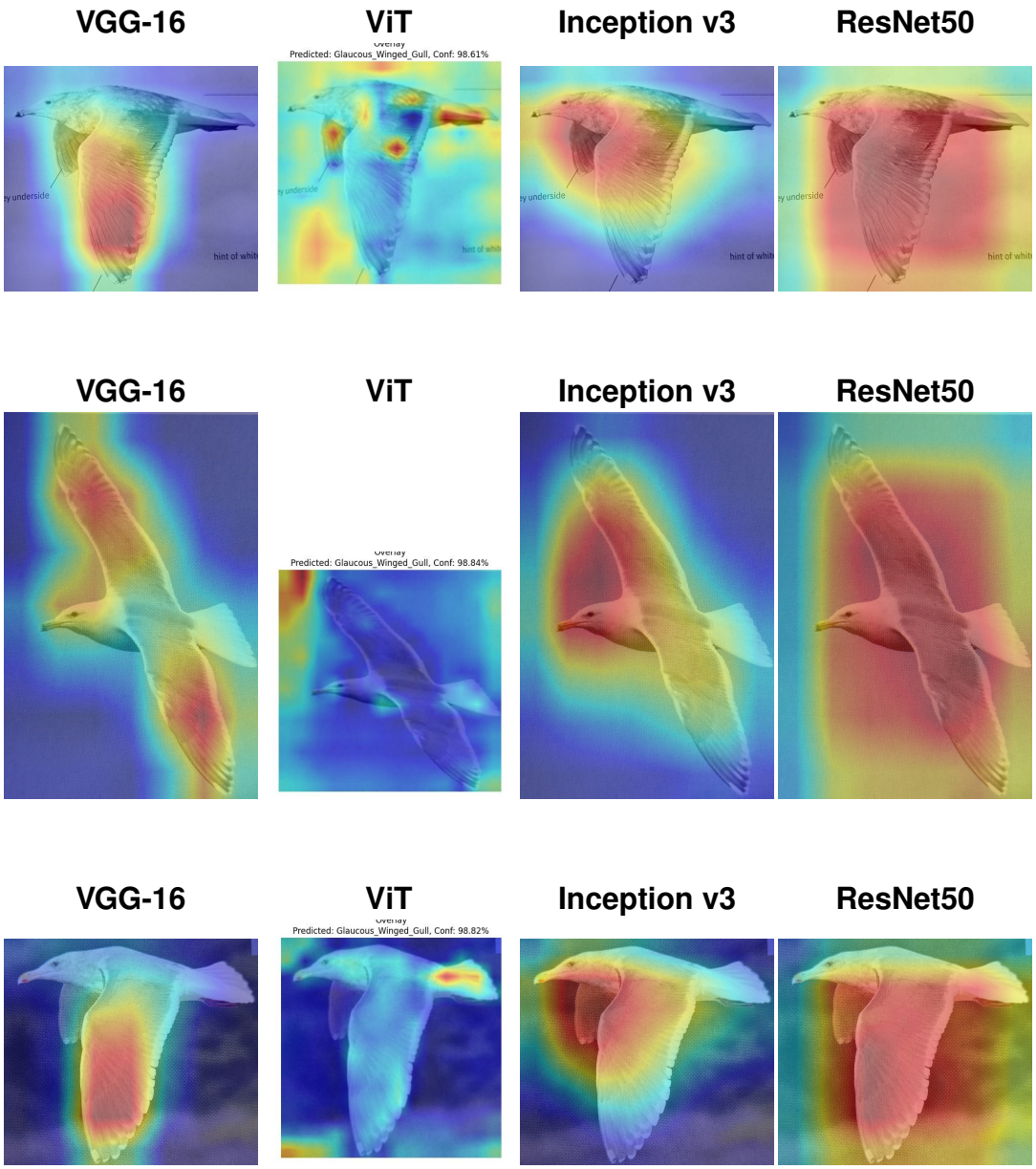
Figure 8.6: ResNet50 model performance visualization

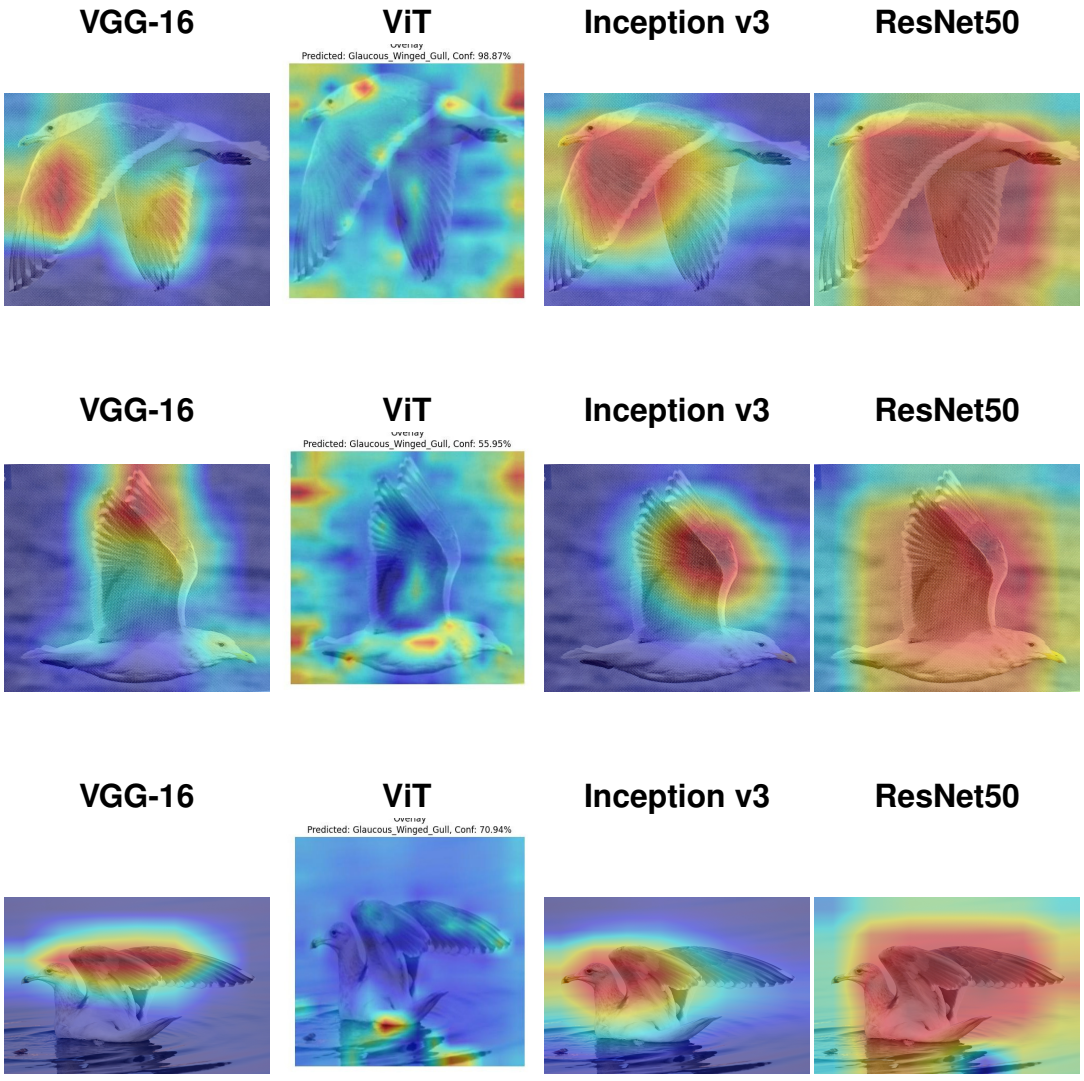
8.2 Model Interpretability Results

8.2.1 Model Comparison with Grad-CAM results on unseen Test Set

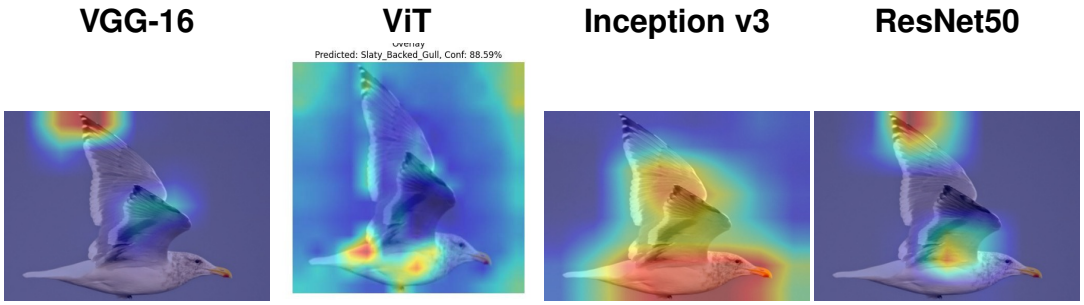
This section presents the results of Grad-CAM across different models except for ViT which displays the attention rollout results. All the images here were correctly predicted by all the well-performing models below. Each row displays the same image processed by the models that achieved high test and validation accuracy, allowing direct comparison of how each model focuses on different bird features.

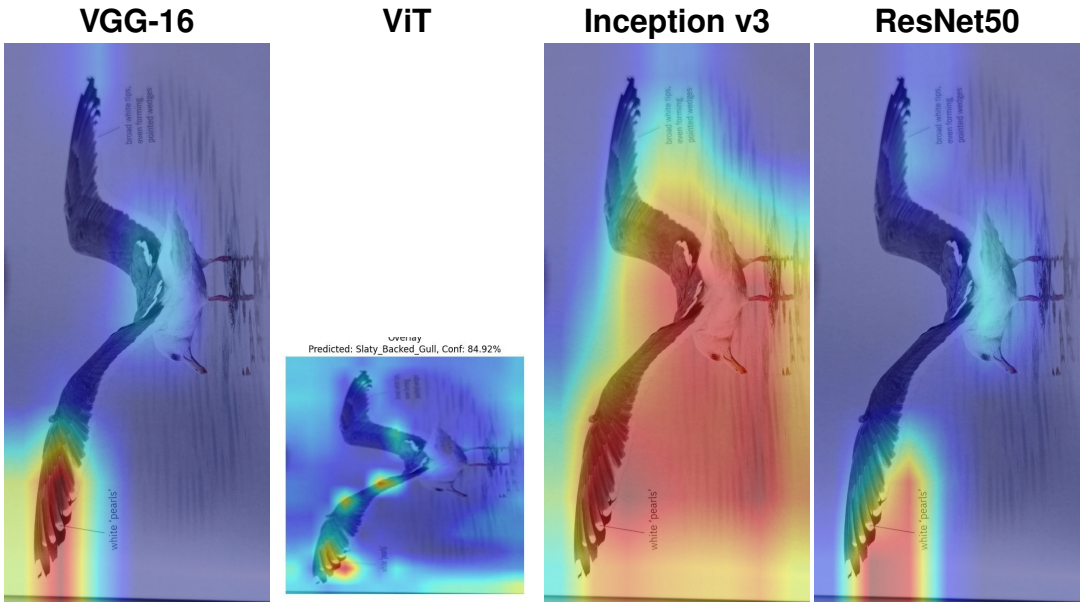
Glaucous Winged Gull Interpretability Results

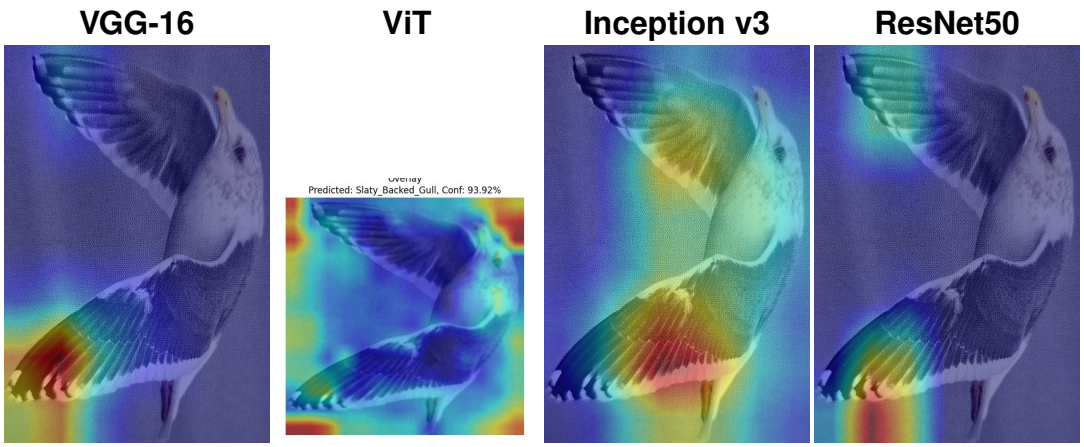


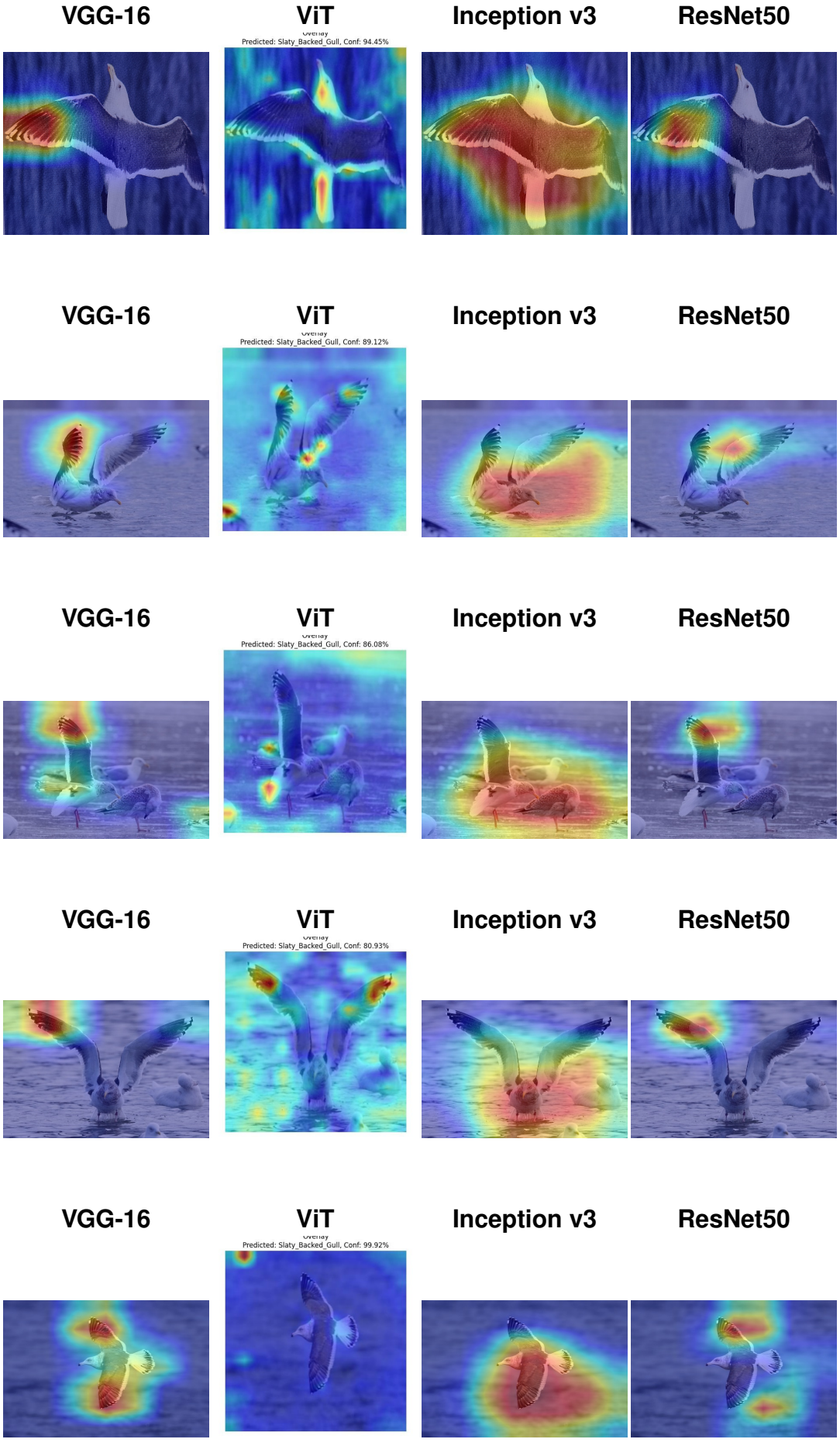

VGG-16
ViT
Inception v3
ResNet50
Predicted: Glaucous_Winged_Gull, Conf: 98.87%
VGG-16
ViT
Inception v3
ResNet50
Predicted: Glaucous_Winged_Gull, Conf: 55.95%
VGG-16
ViT
Inception v3
ResNet50
Predicted: Glaucous_Winged_Gull, Conf: 70.94%

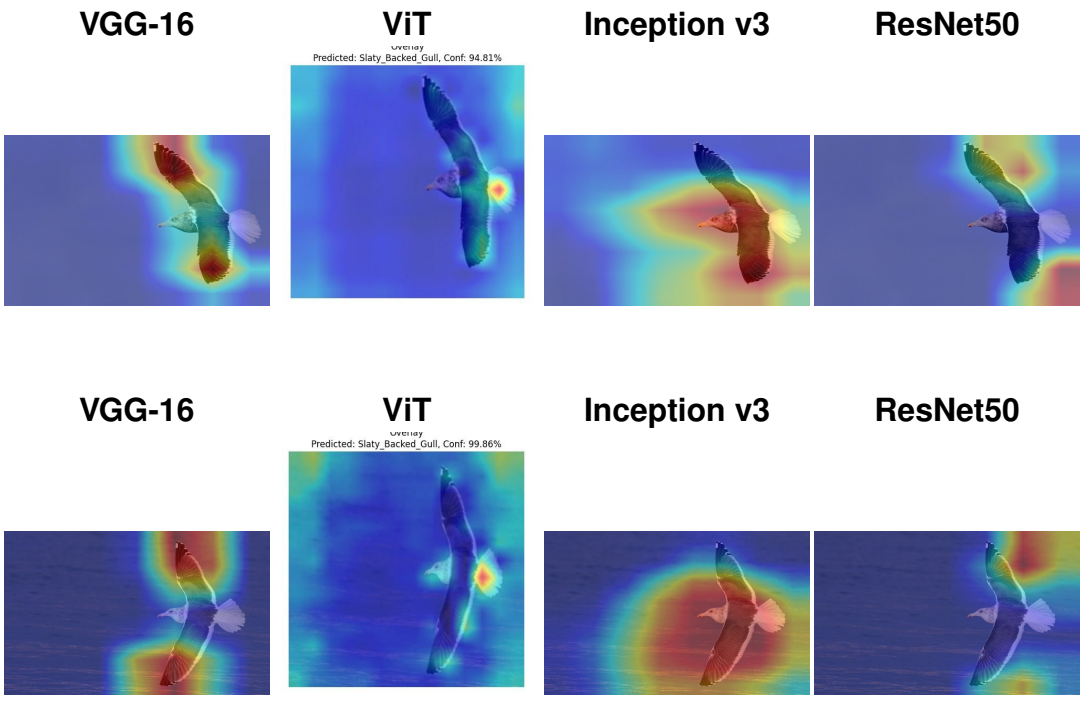
Slaty Backed Gull Interpretability Results



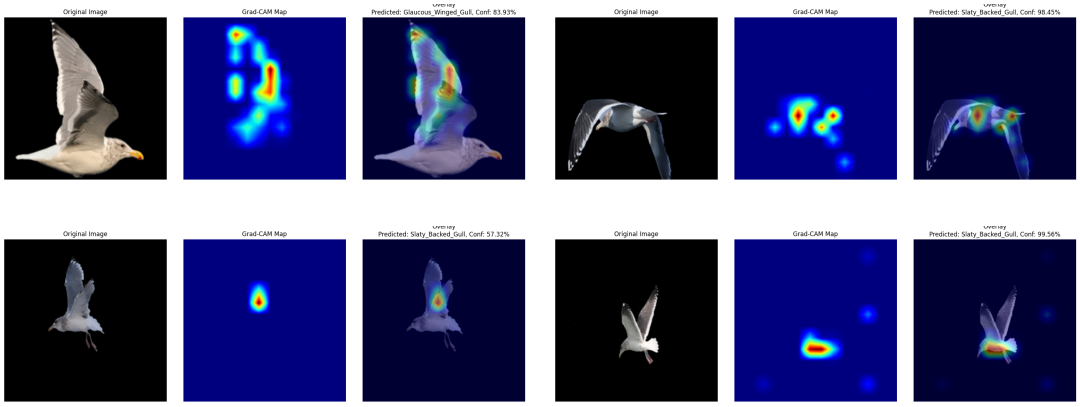




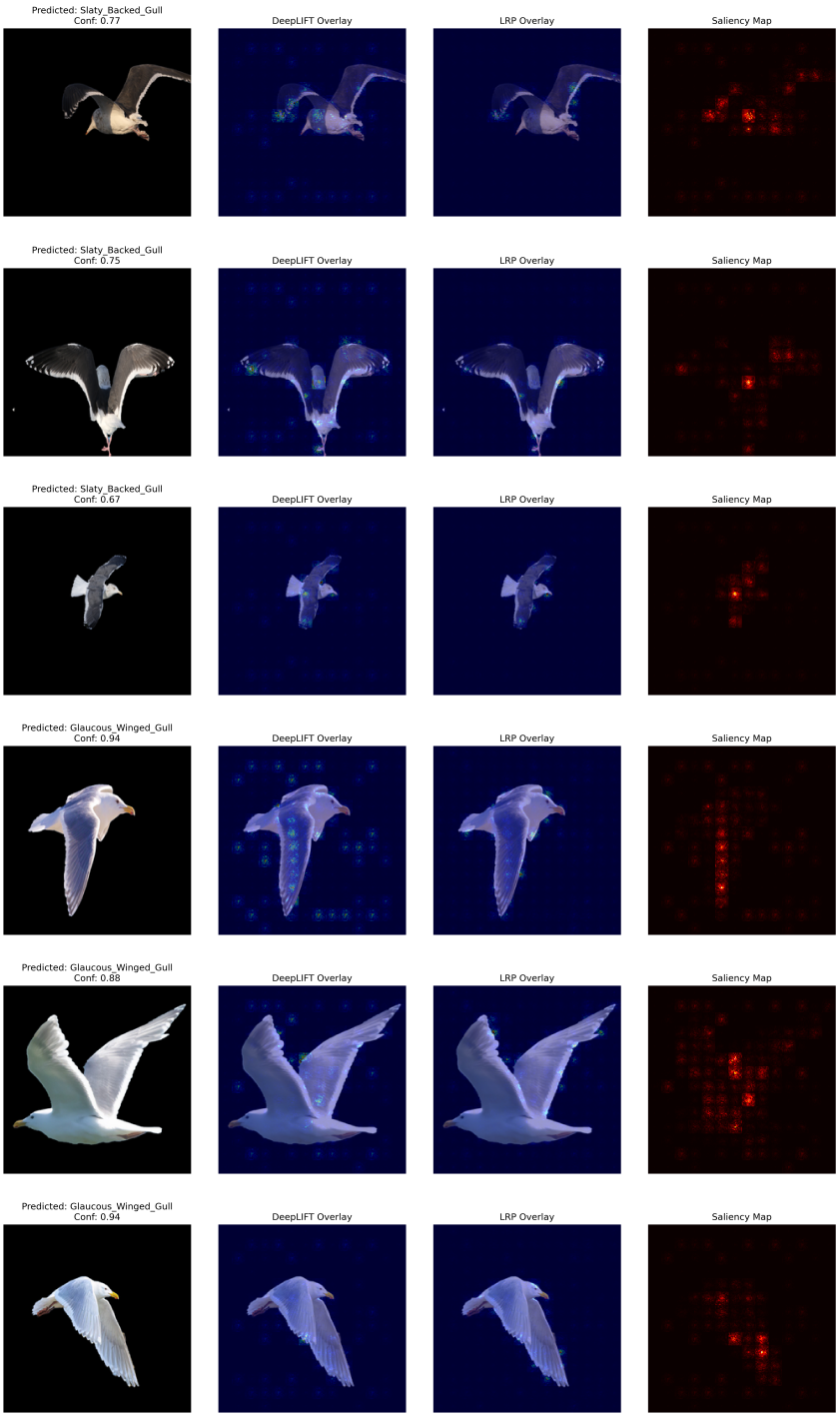

VGG-16
ViT
Inception v3
ResNet50
Predicted: Slaty_Backed_Gull, Conf: 94.45%
VGG-16
ViT
Inception v3
ResNet50
Predicted: Slaty_Backed_Gull, Conf: 89.12%
VGG-16
ViT
Inception v3
ResNet50
Predicted: Slaty_Backed_Gull, Conf: 86.08%
VGG-16
ViT
Inception v3
ResNet50
Predicted: Slaty_Backed_Gull, Conf: 80.93%
VGG-16
ViT
Inception v3
ResNet50
Predicted: Slaty_Backed_Gull, Conf: 99.92%



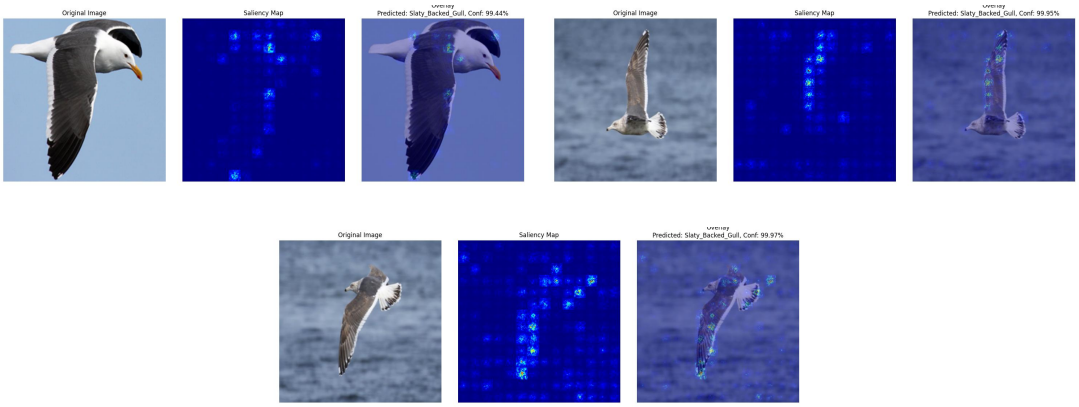
8.2.2 InterpretableViT Interpretability Results



8.2.3 EnhancedViT Interpretability Results



8.2.4 Saliency Map Results of ViT model



8.2.5 Feature Ablation Result of ViT model

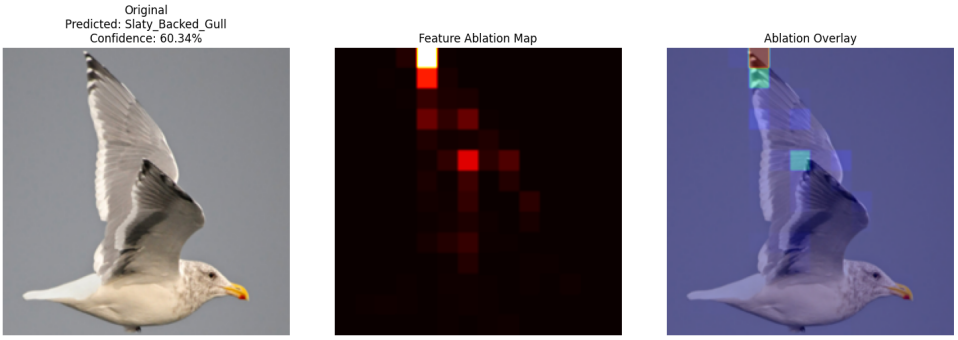
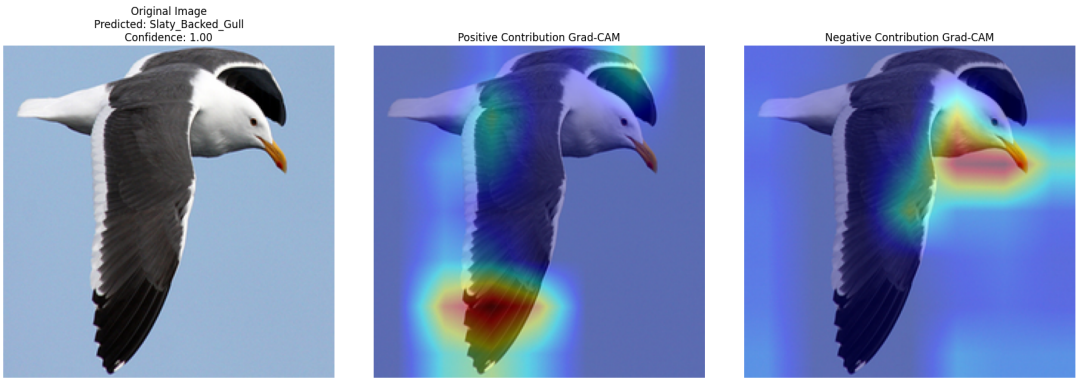
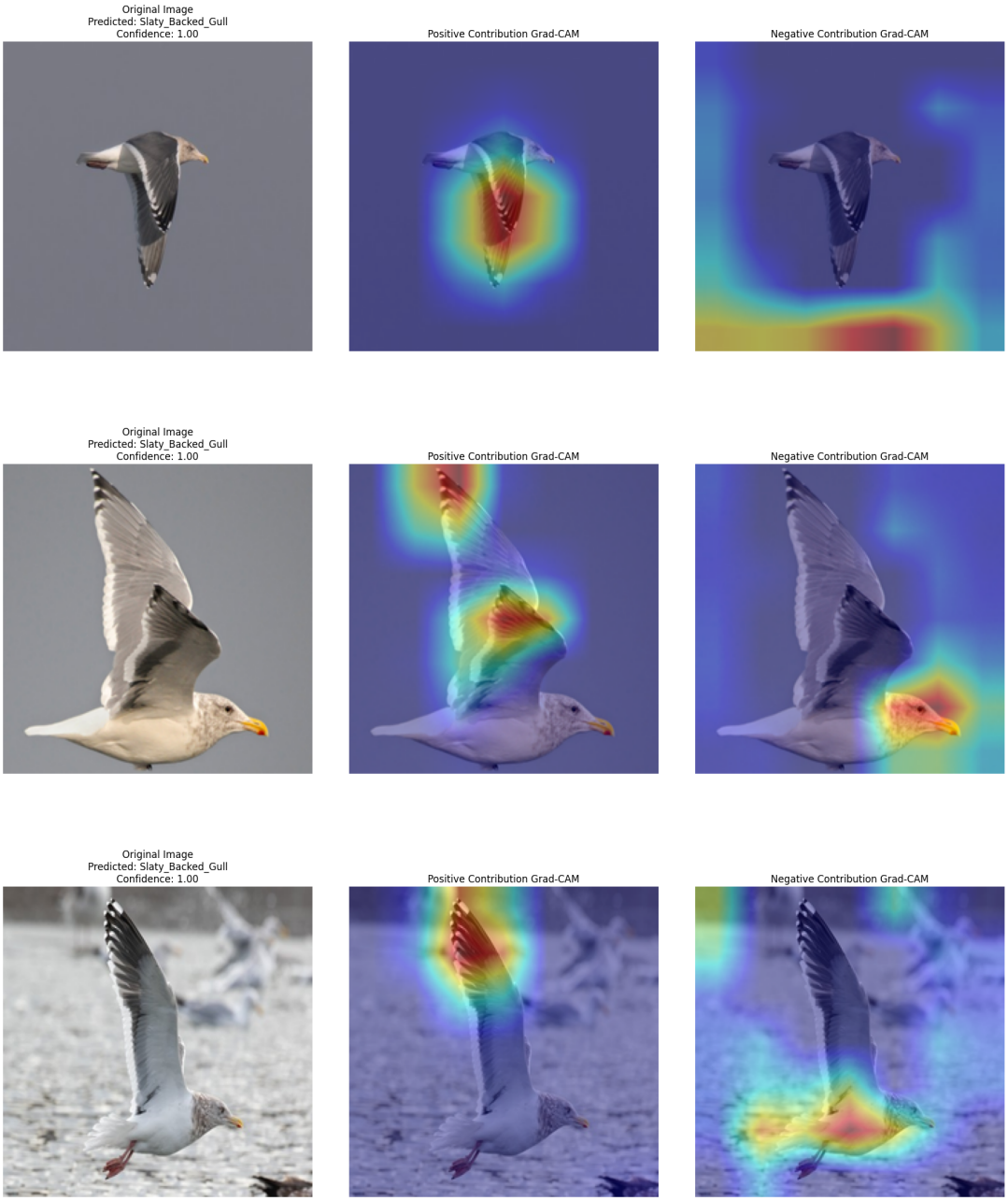


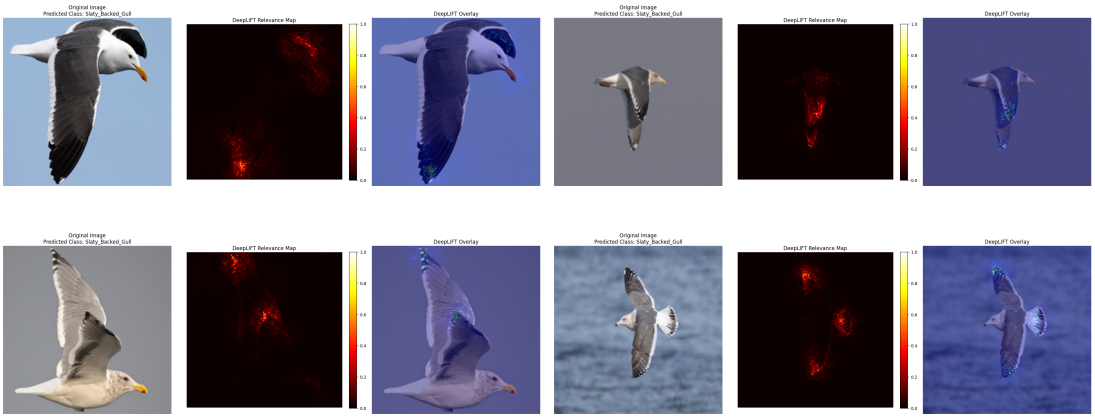
Figure 8.9: Ablation study for ViT

8.3 Results of different Interpretability techniques of VGG Model





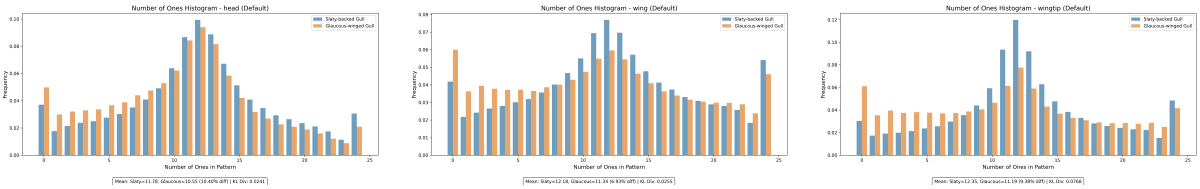
8.3.1 Saliency Map Results of VGG model



8.4 Local Binary Pattern Analysis Results

This section displays the LBP analysis results for the wing, wingtip, and head regions of the gull images.

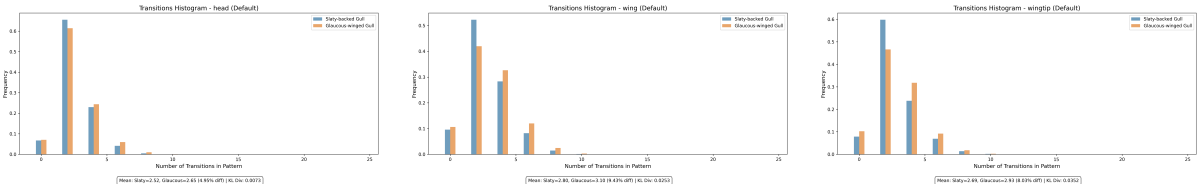
Figure 8.11: Histograms for number of ones



(a) Head region (b) Wing region (c) Wingtip region

Figure 8.12: Local Binary Pattern analysis results showing histograms of pattern distributions across different bird regions.

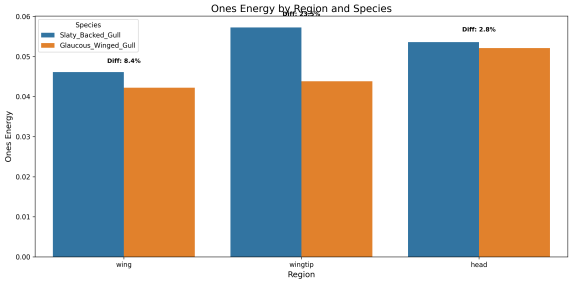
Figure 8.13: Histograms for number of transitions



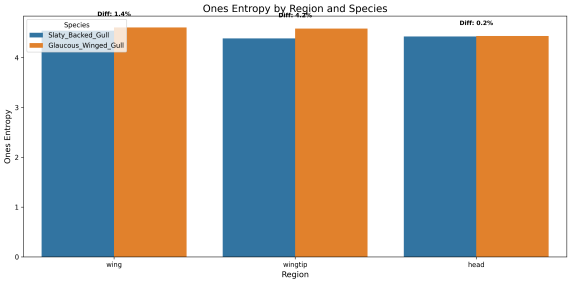
(a) Head region (b) Wing region (c) Wingtip region

Figure 8.14: Local Binary Pattern analysis results showing histograms of pattern distributions across different bird regions.

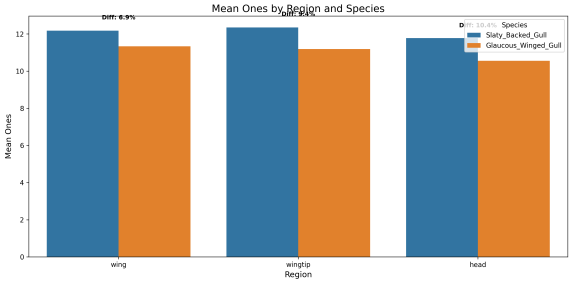
8.4.1 Calculated Texture-based Feature Comparisons on 1s and Transitions Histogram (Rotation Invariant)



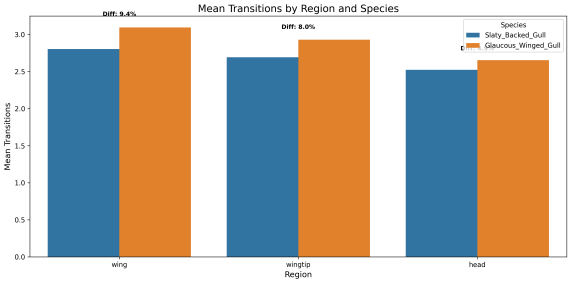
(a) Ones energy comparison



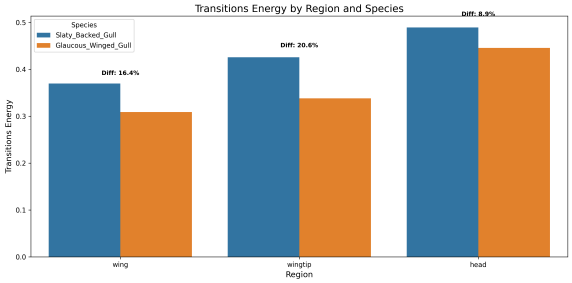
(b) Ones entropy comparison



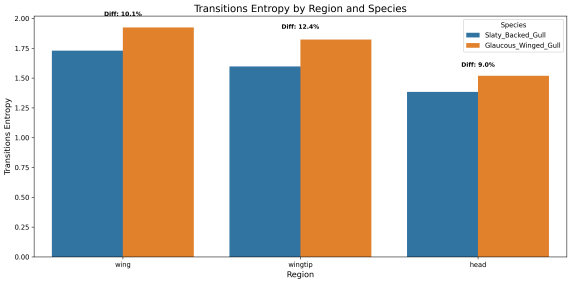
(c) Mean ones comparison



(d) Mean transitions comparison



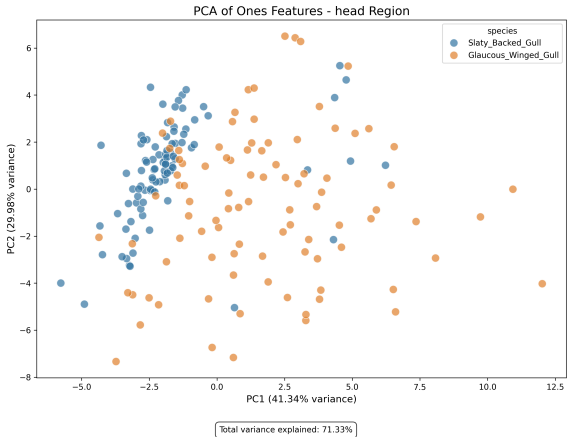
(e) Transitions energy comparison



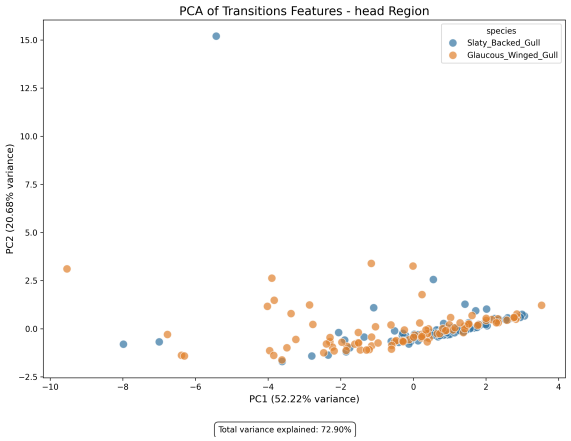
(f) Transitions entropy comparison

Figure 8.15: Selected feature comparison results related to ”ones” and ”transitions” from the original dataset.

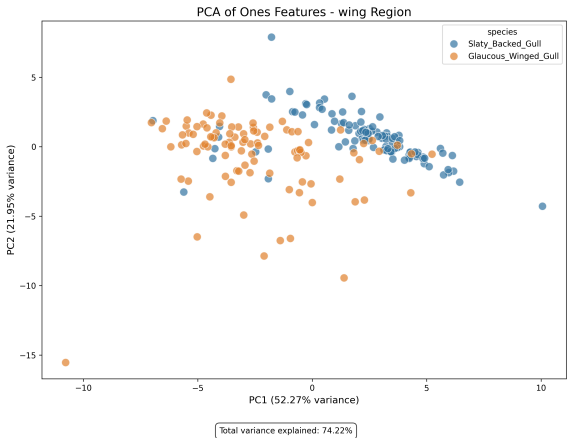
8.5 PCA Plots of 1s and Transitions Histogram



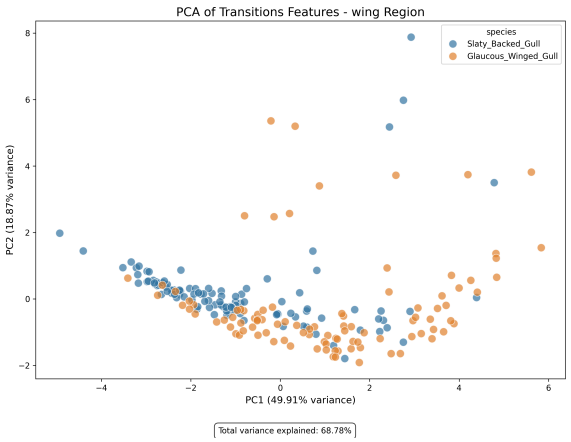
(a) Head ones PCA plot



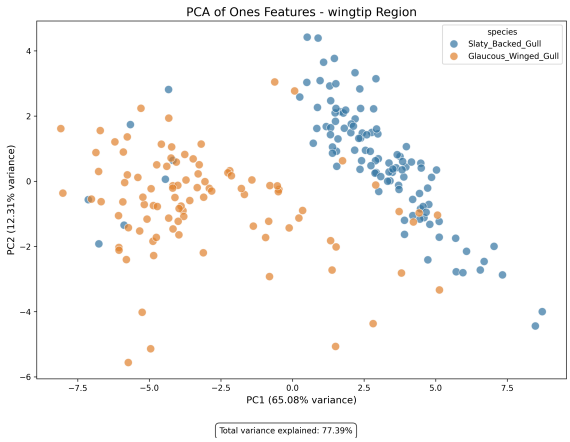
(b) Head transitions PCA plot



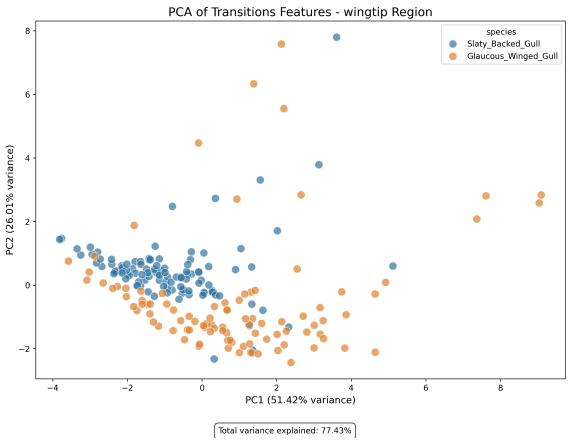
(c) Wing ones PCA plot



(d) Wing transitions PCA plot



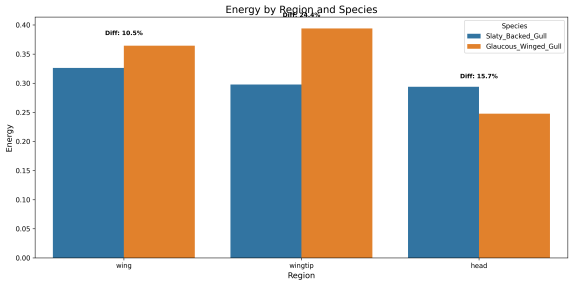
(e) Wingtip ones PCA plot



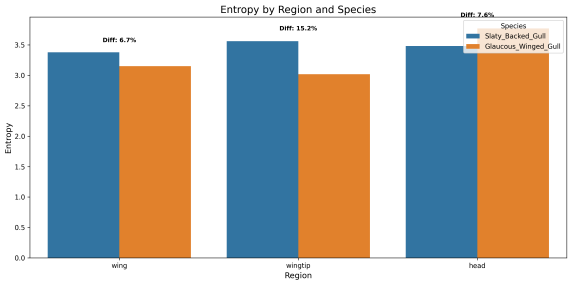
(f) Wingtip transitions PCA plot

Figure 8.16: PCA plots for ”ones” and ”transitions” features across head, wing, and wingtip regions.

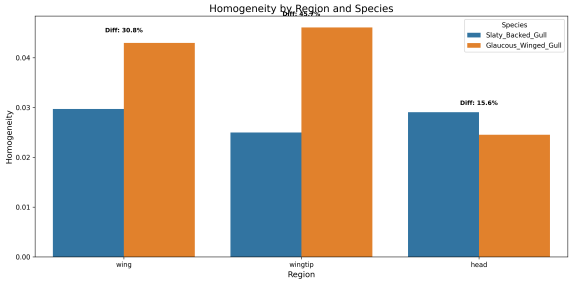
8.5.1 Calculated Texture-based Feature Comparisons on LBP Histogram (May be Rotation Variant)



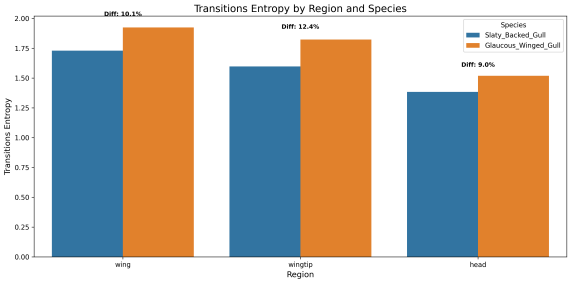
(a) Energy comparison



(b) Entropy comparison



(c) Homogeneity comparison

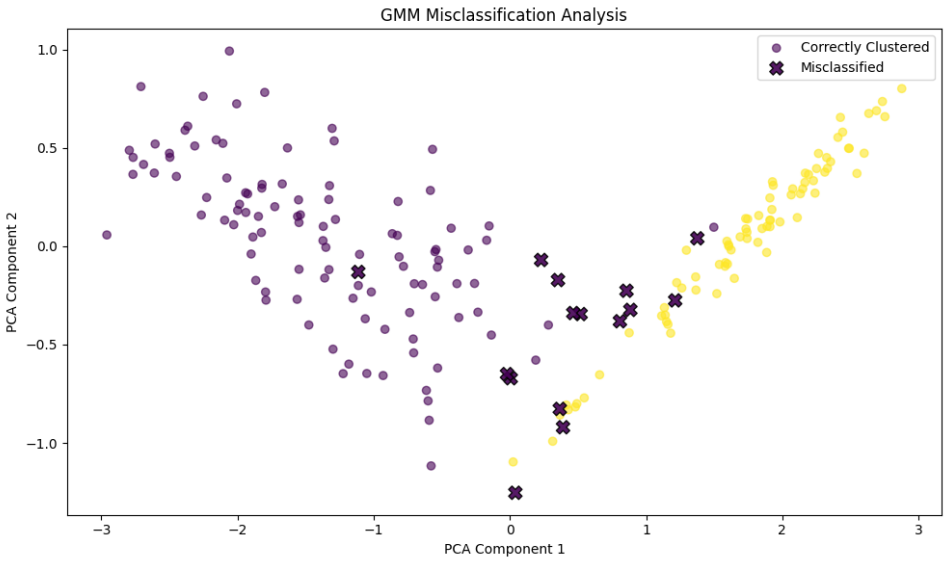


(d) Transitions entropy comparison

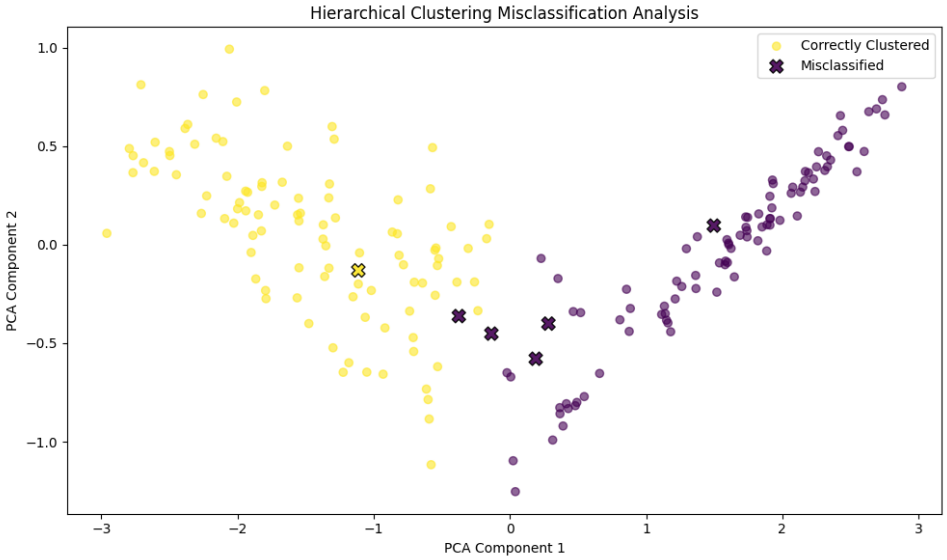
Figure 8.17: Selected feature comparison results from the original dataset.

8.6 Misclassified Points in Clustering Analysis

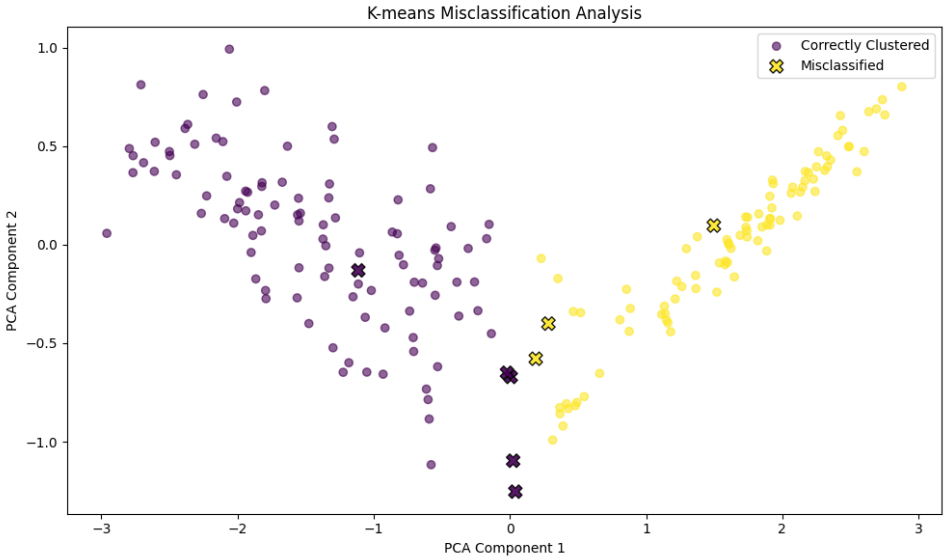
This section presents the misclassified points identified by different clustering algorithms. Each plot highlights the data points that were not correctly grouped according to the ground truth.



(a) Misclassified points using the GMM clustering algorithm.



(b) Misclassified points using the Hierarchical clustering algorithm.



(c) Misclassified points using the K-Means clustering algorithm.

Figure 8.18: Misclassified points by different clustering algorithms.