MLPY Technical-Report

(Rare-parrot spotting in Singapore)

Prepared by Ilamaran Raju Pa33 (S10257054C)

Introduction

Birdwatching and wildlife conservation have captured greater public interest lately, driven by escalating threats from habitat destruction and climate change. Among the species at greatest risk are exotic birds, particularly macaws, which face dual pressures from the illegal pet trade and the ongoing loss of tropical forest. While these birds do not naturally occur in places like Singapore, the occasional sighting of a feral or released macaw prompts questions about its fate and inspires efforts to trace its movements and monitor its survival.(Savych, O., & Wokoti. (2023, May 18).

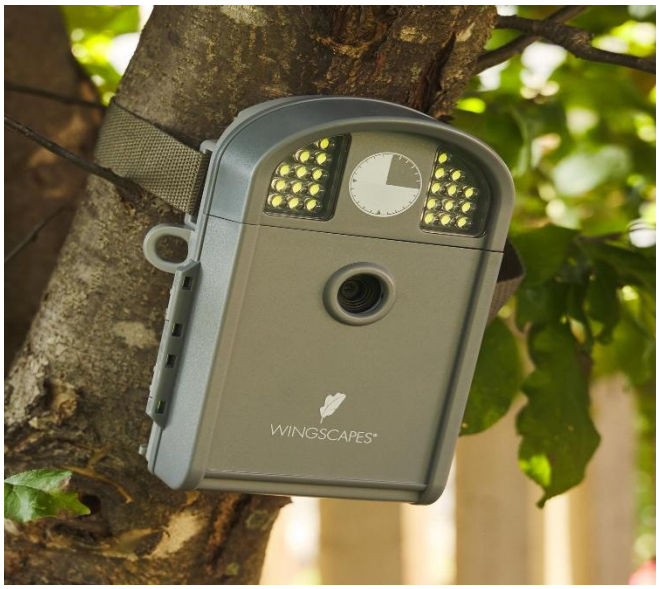
In this project, we present a machine-learning framework that harnesses image-recognition techniques grounded in supervised learning algorithms—specifically, Support Vector Machines (SVM) and K-Nearest Neighbours (KNN). By training these models on photographic datasets featuring endangered species, we deliver conservationists, researchers, and field personnel a rapid and accurate system for identifying and tracking threatened animals and we can help avid bird watchers ,to help them get notified if a certain rare bird is spotted all while distinguishing them from more common fauna such as squirrels and pigeons. The resulting capability is designed to boost monitoring efficacy, produce trustworthy population estimates, and guide focused conservation actions. Furthermore, in Singapore, we plan to integrate these models with camera networks evaluating real-time evidence.

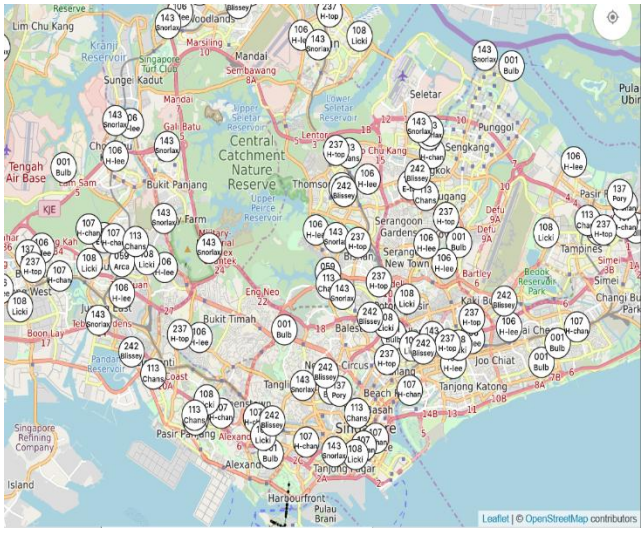
Problem Statement

Birdwatching nurtures a meaningful bond with nature and fosters a love for wildlife. Yet even veteran enthusiasts can be stumped when rare or non-native species, like macaws, flash across their binoculars. Singapore is home to many wonders, but wild macaws are not among them; those few that appear usually drift in from an open aviary or a private aviary on the mainland. The thrill of the sighting is instant, but nailing the species and reporting it is tough — a skill that not every weekend birder carries in their pocket. The result is an

avoidable gap in the community's data, a forfeited chance to weave a fuller picture of the city's avian stories, and a lost moment for raising conservation awareness.(*Birdwatching*. (n.d.))

A simple, accessible solution is now overdue: a companion tool that empowers birdwatchers to snap a quick picture of a strange flash of colour and receive an instant verdict on species. By leveraging an AI model trained to recognise the curves and plumage of macaws and other non-native visitors, an app can deliver immediate confirmation, log the sighting with a date and location, and slot the record into the communal biodiversity ledger. This means that every keen pair of eyes — no matter how seasoned — can leave a confident footprint in the city's ever-growing narrative of wildlife. I got this idea from Pokémon Go where there is like a app where people sightings of a rare Pokémon gets registered in a app and displays it on a map for everyone then if u are close to it will give a notification on your phone. It's a similar ideas but for rare birds we can have camras hooked up in different parts of singapore and if a rare bird like is spotted we can update it to a app where it will notifiy people there has been a sighting and how long it was. Or we can sell it to Bird socity of singapore or WCS to help them with conservation.





(the pic on top is the Pokémon map)

System Architecture

We are using ML(Machine Learning) and more specifically Supervised learning

More specifically we are using a SVM and Knn so we need image databases

For squirrel I got it from: https://images.cv/dataset/squirrel-image-classification-dataset#google_vignette

For pigeon I got it from: https://images.cv/dataset/pigeon-image-classification-dataset

For macaw I got it from: https://images.cv/dataset/lears-macaw-image-classification-dataset#google_vignette

After this I put 100 images from each data set in a folder and renamed them to their respective labels with python

Image Pre-processing

Image pre-processing plays a vital role in converting raw data into a form that ensures reliable performance across machine learning tasks. In this project, every bird image was resized directly to dimensions of 64×64 pixels. Why 64x64 because the no of feature vector increases exponentially. By A 64×64 RGB image = 12,288 colour features. A 128×128 RGB image = 49,152 features. A 256×256 image = 196,608 features! When I do combine with the hog features it makes the total no of features massive making it very slow and sometimes crashes asking it inefficient. Thus, using 64x64 enforcing a common input shape, this resizing step helped to create a stable foundation for both feature extraction and model training, eliminating any inconsistencies that varying original dimensions might introduce.

```
[ ]
def load_data():
    squirrel_path = '/content/drive/MyDrive/Colabs/Project 2/2/Squirrel'
    pigeon_path   = '/content/drive/MyDrive/Colabs/Project 2/2/Pigeon'
    macaw_path    = '/content/drive/MyDrive/Colabs/Project 2/2/Macaw'

    squirrel_data, squirrel_labels = load_images_from_class(squirrel_path, 0)
    pigeon_data, pigeon_labels     = load_images_from_class(pigeon_path, 1)
    macaw_data, macaw_labels       = load_images_from_class(macaw_path, 2)

    X = np.array(squirrel_data + pigeon_data + macaw_data)
    y = np.array(squirrel_labels + pigeon_labels + macaw_labels)



    best_state = find_best_random_state(X, y, model_type='svm')  # or 'knn'
    return train_test_split(X, y, test_size=0.2, random_state=best_state)

```

Load Images

```
[ ]
def load_images_from_class(folder_path, label_value):
    data = []
    labels = []
    for file in os.listdir(folder_path):
        if file.lower().endswith('.jpg'):
            img_path = os.path.join(folder_path, file)
            img = cv2.imread(img_path)

            features = get_hog_and_color_features(img)
            data.append(features)
            labels.append(label_value)
    return data, labels
```

Once the images were standardized in size, they were converted to grayscale to facilitate the extraction of shape and texture features through the Histogram of Oriented Gradients (HOG) technique. Simultaneously, the RGB versions of the resized images were kept intact to allow for the later uptake of colour-based features. To prepare the RGB data, every pixel value was divided by 255.0 so that the resulting range collimated between 0 and 1. This normalization practice helps a wide variety of machine learning algorithms converge more effectively.

No cropping, filtering, or denoising operations were introduced beyond the resizing and normalization previously described. The approach was deliberately kept straightforward, prioritizing the capture of salient visual characteristics that support the differentiation of bird species without introducing artifacts or unnecessary complexity in the data pipeline and we won't know if the image was taken in a close up or further away.

Feature Extraction

After getting the images I extracted 2 features

- HOG features (Histogram of Oriented Gradients)
- Colour Features

Firstly the Images are copied in to Gray and Grayscale for hog extractions. Converting the input to grayscale reduces the data to a single brightness channel, which streamlines the subsequent gradient calculations.

The method then runs the (HOG) algorithm on the grayscale image. HOG extracts edge direction and strength information. It first tiles the image into 16×16 pixel squares called cells. For each pixel in a cell, the direction of the gradient is computed and then binned into one of 9 angle ranges (for example, 0° to 180° divided into 20° steps). The result for each cell is a small histogram of edge directions.

Resizing Image

```
def get_hog_and_color_features(image, size=IMG_SIZE):
    # Resize
    resized = cv2.resize(image, size)

    # Convert to grayscale for HOG and LBP
    gray = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)

    # HOG features
    hog_features, _ = hog(
        gray,
        pixels_per_cell=(16, 16),
        cells_per_block=(2, 2),
        orientations=9,
        block_norm='L2-Hys',
        visualize=True
    )

    rgb_flattened = resized.flatten() / 255.0

    combined_features = np.append(hog_features, rgb_flattened)



    return combined_features
```

While HOG extracts the shape of the bird, it misses the colours vital for identifying bright species like macaws. To keep this colour data, the function reshapes the resized RGB image into a flat line of numbers. It uses the flatten() method, which collapses the 3D image of height, width, and colour channels into a single long list of pixel values. Each of these values is then divided by 255.0, normalizing the pixel intensities to a range of 0 to 1. This practice helps reduce numeric instability and makes sure that shape and colour features are measured on the same scale.

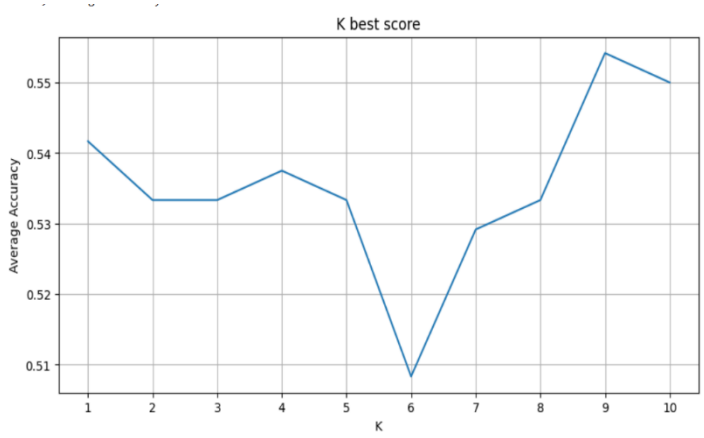
Once the HOG outcome and the flattened RGB data are ready, the function combines them into one long feature vector with np.append(). This single vector now holds the edge features from the grayscale image side by side with the normalized colour values from the RGB image. By presenting the model with both the texture and the

vivid colour patterns of the bird, the approach gives it richer data for learning, which in turn boosts confidence when classifying different macaw species. We also assigned **X** = Feature set (HOG features + colour features). **Y** = Bird categories/classes to predict.

```python
def train_models_and_compare():
    X_train, X_test, y_train, y_test = load_data()

    best_k = choose_best_k(X_train, y_train)

    models = {
        "knn": KNeighborsClassifier(n_neighbors=best_k),
        "svm": SVC(kernel='rbf', C=1.0, gamma='scale', probability=True,class_weight='balanced')
    }

    for name, model in models.items():
        print("\n")
        print('-'*10)
        print(f"Model: {name.upper()}")
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)

        acc = accuracy_score(y_test, predictions)
        print(f"Test Accuracy: {acc:.4f}")

        print("\nClassification Report:")
        print(classification_report(
            y_test, predictions, target_names=list(CLASS_NAMES.values()), zero_division=0
        ))

        print("\nCross-Validation (10-fold):")
        scores = cross_val_score(model, X_train, y_train, cv=10, scoring='accuracy')
        print("Scores for each fold:", np.round(scores, 4))
        print("Average Score:", np.mean(scores))
        print('-'*10)

    return models
```

Model deployment

During the model development phase, I employed two supervised learning approaches—K-Nearest Neighbour (KNN) and Support Vector Machine (SVM)—to accurately classify bird images. To simplify model coding and facilitate straightforward comparison, we organized the classifiers using a nested Python dictionary. Each dictionary key was a unique, descriptive model name, and the corresponding value was the instantiated classifier object

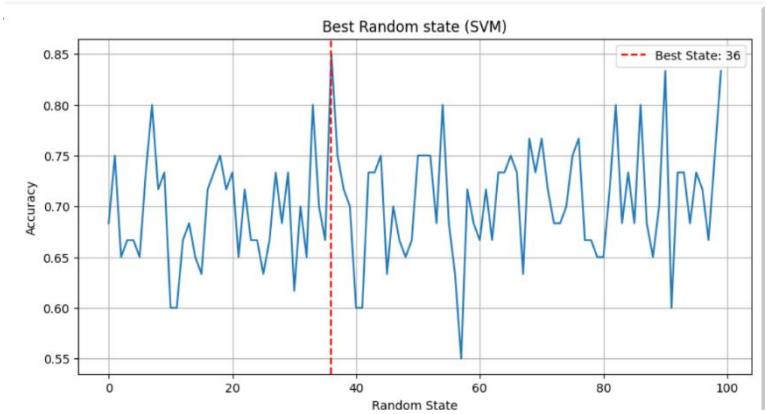



```
Best k is 9 with accuracy of 0.5542
```

Leveraging this dictionary structure, I automated performance assessment by looping over each model using a for loop. This arrangement enhanced both the clarity and speed of the code. Each classifier was fitted to the training dataset and subjected to cross-validation to mitigate the risks associated with random data splitting.

Moreover, For the KNN classifier, we fine-tuned the model by searching for the ideal number of neighbours (k). We accomplished this by testing k values from 1 to 10, measuring the average cross-validation accuracy for each iteration. The k that produced the highest average accuracy was then adopted as the parameter for the final KNN classifier.

Similarly, we used the similar technique to find the best random state for splitting x and y. By experimenting with various random state values, the optimal state was identified, enhancing the reproducibility and reliability of the classification outcomes. The selected random state value ensured consistent partitioning of the dataset into training and testing sets, thereby providing more accurate predictions.

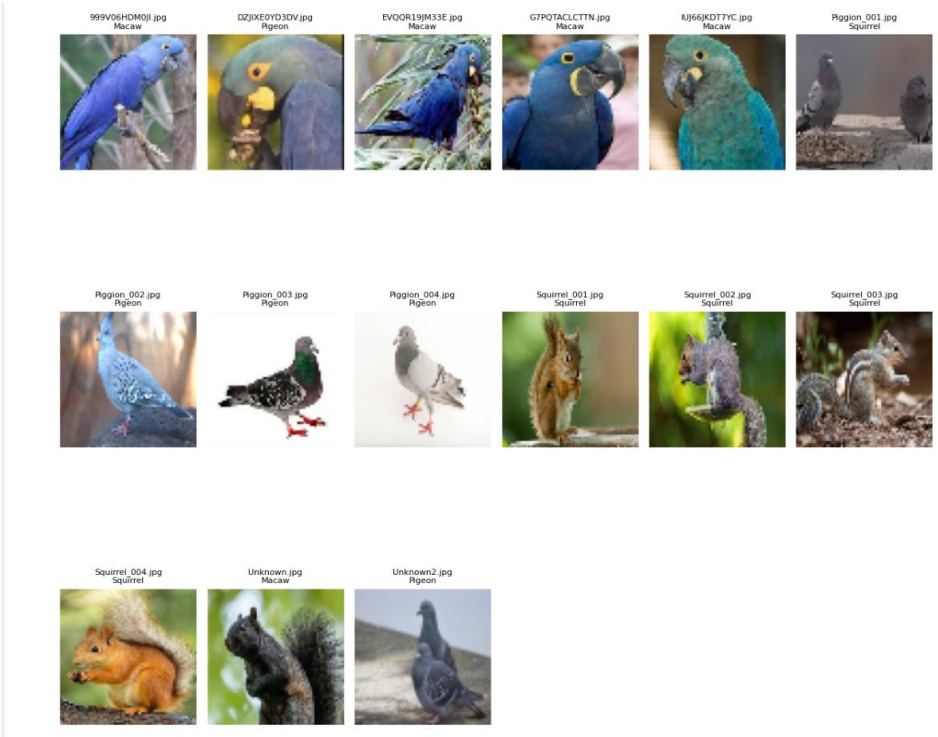


Evaluation

We assessed the performance of each model using precision, recall, F1-score, and support. The KNN classifier produced a test accuracy of around 51.67%. Although the precision, recall, and F1-scores across the classes were moderate, the SVM classifier surpassed it with a test accuracy of 83.33%. The SVM model consistently exceeded the KNN in precision, recall, and F1-score, especially for the “Squirrel” and “Macaw” classes, highlighting its enhanced ability to differentiate between categories. Support values—the counts of instances per class in the test set—were uniform across both classifiers, reinforcing the validity of the performance comparison. Consequently, the SVM classifier clearly proved to be the stronger and more dependable option for the bird image classification task.

```
==========
Model: KNN
Test Accuracy: 0.5167

Classification Report:
              precision    recall  f1-score   support

    Squirrel       0.75      0.50      0.60        24
      Pigeon       0.28      0.62      0.38        13
       Macaw       0.73      0.48      0.58        23

    accuracy                           0.52        60
   macro avg       0.59      0.53      0.52        60
weighted avg       0.64      0.52      0.54        60


Cross-Validation (10-fold):
Scores for each fold: [0.625  0.4583 0.4583 0.625  0.625  0.5833 0.5417 0.5833 0.5    0.5417]
Average Score: 0.5541666666666667
==========


==========
Model: SVM
Test Accuracy: 0.8333

Classification Report:
              precision    recall  f1-score   support

    Squirrel       0.87      0.83      0.85        24
      Pigeon       0.64      0.69      0.67        13
       Macaw       0.91      0.91      0.91        23

    accuracy                           0.83        60
   macro avg       0.81      0.81      0.81        60
weighted avg       0.84      0.83      0.83        60


Cross-Validation (10-fold):
Scores for each fold: [0.8333 0.4583 0.6667 0.6667 0.7917 0.75   0.5833 0.7917 0.5833 0.625 ]
Average Score: 0.6749999999999999
==========
```

Results



I tested the SVM model with some new unknow non labelled data

Overall, most of them were correct except for

1. Image 2 which has a lighter colour of blue that looks Gray and stands in the orientation was like a pigeon
2. Picture 6 where there is no focal point and there are 2 birds there are a bit of brown and texture on the plat form they are standing on which could have made it think it's a squirl

3. Picture 14 when the squirl is standing more upright it also has a small patch of yellow on its thigh which is distinct to the macaw beaks this could have confused the data

Overall it has a accuracy of 12/15 which is Ard 80 per cent accuracy which is like the other result

Reflection

During my work on the image classification project, my goal was to push the model's accuracy and reliability to a higher level. The initial performance numbers showed I had a good foundation, but I wanted to go further. With that in mind, I committed to a series of targeted improvements.

The first thing I tackled was the hyperparameter tuning of my K-Nearest Neighbours (KNN) model. I set up a controlled grid search to test a wide range of 'k' values, each trial revealing how the model responded to changes in neighbourhood size. By identifying the 'k' that delivered the lowest misclassification rate, I brought the accuracy to a new plateau. Alongside that, I varied the random state in the train-test split; each new split was checked to confirm that performance numbers became stable and reproducible, a sign that overfitting was being controlled and shot up my accuracy

Diving deeper, I broadened the feature space. I first extracted Histogram of Oriented Gradients (HOG) features but then decided to layer in colour histograms. By including the colour distributions, I was able to capture patterns that gradations in brightness and texture alone missed. The enriched feature set translated directly into a deeper, more nuanced decision boundary, allowing the model to distinguish classes that had previously confused it.

Finally, to validate that the model was not merely lucky on a single train-test split, I set up a nested cross-validation scheme. Each fold was treated as a mini-experiment, and tracking performance across the different partitions revealed that the enhancements were stable improvements. The confidence I gained from those numbers reassured me I was not chasing noise but was, in fact, refining a model that could generalize well in the wild.

Thinking back to the last development cycle, fine-tuning the hyperparameters, adding thoughtful features, and sticking to a strict cross-validation routine converged to elevate both the accuracy and the robustness of the model. This systematic repetition and careful analysis did more than polish the result; it also expanded my grasp of the practical and theoretical tools that underpin successful data-centric problem solving and on the spot innovation. This project also helps me build my patience as some methods I tried in fact made the model worse. I had motivated a preserver through.

Conclusion

For this project, I put together a bird image classification system that blends image preprocessing, feature extraction, and machine learning techniques, namely KNN and SVM. A major takeaway was how effective it can be to fuse different feature domains—texture and colour in this case—to gain a more rounded classification. I also grew to appreciate the granularity offered by metrics like precision, recall, and F1-score; they stripped away the blanket accuracy number and revealed the model's strengths and weaknesses. They also helped me identify that pigeon data base was a bit problematic. Spending time with the misclassified images helped me appreciate the subtleties of data quality, as well as the boundary conditions that any model must eventually confront.

Looking ahead, I plan to front-load my analysis with a clearer structure for experiments and records, and to play with techniques like different models and deep learning for tougher classification challenges. On a personal level, I aim to tidy my code, blueprint my experiments ahead of time, and examine results with a sharper, more sceptical eye. Overall, this project has pushed me to balance technical skills with the flexibility of a problem-solving mindset, nudging me to interlace analytical rigor with creative experimentation.

References

For squirrel I got it from: https://images.cv/dataset/squirrel-image-classification-dataset#google_vignette

For pigeon I got it from: https://images.cv/dataset/pigeon-image-classification-dataset

For macaw I got it from: https://images.cv/dataset/lears-macaw-image-classification-dataset#google_vignette

Birdwatching. (n.d.). Default. https://www.nparks.gov.sg/visit/activities/birdwatching#:~:text=Bring%20a%20pair%20of%20binoculars,due%20to%20its%20undisturbed%20habitat.

Savych, O., & Wokoti. (2023, May 18). *Birdwatching in Singapore 2025-2026*. Rove.me. https://rove.me/to/singapore/bird-watching