# Bird Species Classification Comparative Analysis: Machine Learning and Deep Learning Approaches

Max Li

Thompson Rivers University

DASC 6810 Data Science Seminar

Dr. Mohamed Tawhid

November 20, 2023

## Abstract

Birds are an essential component of the ecosystem. Identifying and classifying different types of bird species in the wilderness is pivotal in biodiversity monitoring and ecological research. In this project, we completed an analysis in order to compare traditional machine learning algorithms and deep learning techniques for bird species classification using an image dataset acquired from Kaggle. For machine learning methods, we employed logistic regression, Random Forest, and Support Vector Machines (SVM) to delineate bird species based on image features extracted using a pre-trained VGG16 model. For the deep learning approach, the ability of Convolutional Neural Networks (CNN) was exploited in recognized complex patterns. Both approaches were evaluated with varying class sizes—precisely 10 and 100 classes. The methodical approach involves the preprocessing of data, followed by feature extraction for machine learning models and direct input to the deep learning model. Both methodologies yield high accuracy rates, with the deep learning approach demonstrating better performance in capturing the nuanced features critical for accurate bird species identification, especially with a more significant number of classes, while maintaining performance integrity. This project also offered valuable views on the scalability of deep learning and explored its potential in processing complex, high-dimensional data in ecological monitoring under the same framework.

*Keywords:* Species classification, Logistic regression, SVM, Random Forest, CNN

## 1. Introduction

In recent years, the application of deep learning techniques in ecological research has significantly grown due to their higher performance and flexibility. Such advanced computational techniques in neural networks have been employed to automate the processing of data collected in various forms, such as camera-trap images, audio recordings, and video footage (Depauw et al., 2022). This data processing automation is particularly impactful in ornithological research, where the neural network models can learn from the data and extract optimal features for solving classifying problems(Sarker, 2021). Therefore, the neural network models are helpful in species classification from pictures, where numerous features are to be considered and analyzed (Elhamond et al.). It is also true that for the past decades, traditional machine learning algorithms such as Logistic Regression, Random Forest, and SVM have been successfully implemented in species classification in ecology(Christin et al., 2019). Image classification algorithms based on each approach possess unique strengths. Examining and evaluating the image classification performances between traditional machine learning and deep learning approaches is crucial for choosing the appropriate algorithm for species categorization under different situations.

In this project, we compared the representative Logistic Regression, Random Forest, SVM, and CNN algorithms with bird species classification accuracy and run time to propose a benchmark for choosing the suitable image classification algorithm. The models were trained using two varying-size image datasets to compare algorithm performance across different data scales. Since the listed model yielded different results in classification experiments, this project also measured the quality of the classification results with other metrics such as loss rates, run time, and overall accuracy.

The study is structured as follows: Section 2 covers required materials and methods, Section 3 details the dataset and classifiers, and Section 4 summarizes the outcomes and performance for bird species classification. The paper concludes with Section 5, where we will also suggest future research directions.

## 2. Materials and methods

This section outlines the workflow of the bird species classification procedure, computational resources, datasets, and algorithms used in conducting the comparative project on image classification. The scalability of both machine learning and deep learning models was also compared with different datasets. The classification procedure is detailed in Figure 1.

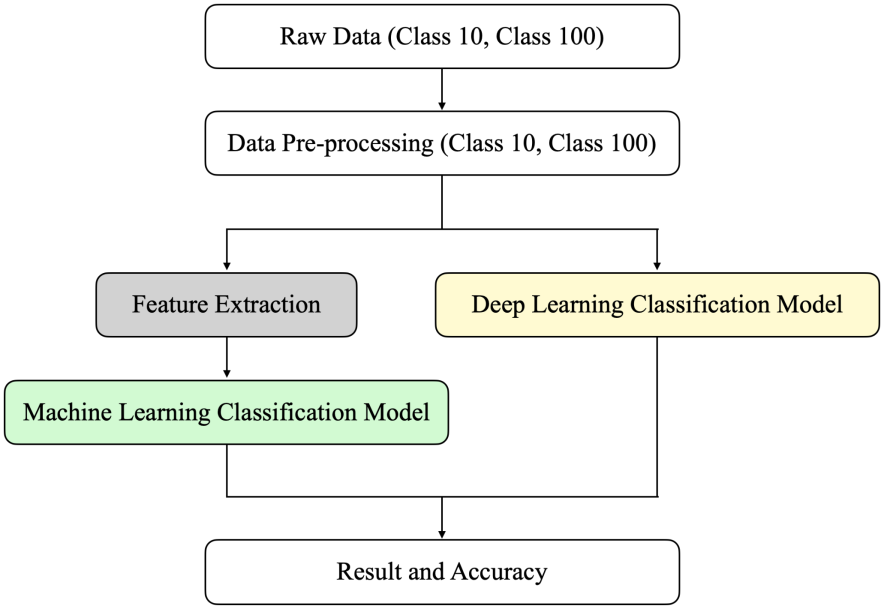


**Figure 1. Procedure workflow of bird species classification (self-painted)**

### *2.1 Computational Resources*

The execution code in this project was written in Python 3.7, utilizing the Google Colab platform for its cloud-based runtime environment with Google's T4 GPU 16GB computing capabilities to ensure a consistent and reproducible environment for all experiments. The machine learning models were executed using the TensorFlow-based Keras library, and the deep learning model was trained with the PyTorch library.

### *2.2 Data Preparation*

Given a set of images $\Upsilon = \{I_1, I_2, \ldots, I_N\}$ where each image $I_i$ is an element in the space of $R^{w \times h \times d}$, with $w$, $h$, and $d$ denoting the width, height, and depth respectively. The depth in here represents the color channels of the input images. Let $I_i^c$ represent the pixel values for channel $c$,

of the image $i$, where $c$ takes values ranging from 1 to $d$. For example, $I_1^1$ denotes all the pixel values in the first color channel of the first input image. Each image $I_i$ is associated with a label $l_i$, where $l_i$ is an element of the set of total species classes $C = \{C_1, C_2, \ldots, C_M\}$, and $M$ is the total number of species classes. The input to the models consists of pairs of images and their corresponding labels, which can be represented as $\{(I_1, l_1), (I_2, l_2), \ldots, (I_N, l_N)\}$.

For the machine learning models, the images are first normalized and resized to fit the input requirements of the VGG-16 network. Subsequently, the VGG-16 model extracts features from the images and outputs a feature vector $f_i \in R^k$ for each image $I_i$, where $k$ denotes the dimension of the output layer. Finally, the resulting feature vectors along with their labels, are stored in a comma-separated values file (CSV) for further machine learning tasks.

For the deep model, each image $I_i$ is directly loaded to be resized and transformed to feed the training process.

### *2.3 Model Architectures*

*2.3.1 VGG-16 for Feature Extraction*: The VGG-16 is a convolutional neural network known for its depth and performance in image recognition tasks (Simonyan et al., 2014). In this project, our machine learning approach applied the pre-trained VGG-16 network to extract image features. These features are represented as multi-dimensional arrays, with each array capturing the essence of the image in terms of learned visual patterns (Tammina, 2019). The multi-dimensional arrays are then flattened and stored in CSV format alongside the corresponding labels of bird species. Finally, the CSV files were input into three machine learning classification models based on Logistic Regression, Random Forest, and Support Vector Machine for further analysis.

*2.3.2 Logistic Regression*: Logistic Regression is a probability-based supervised learning method that is extensively employed for classification tasks. It can predict a continuous numeric output when dealing with regression tasks (Nazish et al., 2021). Logistic Regression uses the relationship between a non-linear explanatory variable to predict the outcome of a discrete response variable. The sigmoid function is applied to assign probabilities that fall within a range from 0 to 1.

*2.3.3 Random Forest*: A Random Forest classifier employs an ensemble learning approach for both classification and regression purposes, constructing numerous decision trees during the training phase. It introduces additional randomness to the model while growing the trees and then combines the results from different trees to produce a more stable and accurate prediction. The Random Forest classifier is effective against overfitting as it averages the predictions from all trees to improve the prediction accuracy.

*2.3.4 Support Vector Machines*: Support vector machines (SVM) are a set of supervised machine learning methods for classification, regression, and outliers detection tasks. They function by recognizing data points in hyper-planes in high-dimensional space, which is especially effective in distinguishing data points from different categories. The data points that are nearest to the hyper-planes are defined as support vectors, which are critical in determining the margin between different classes. These support vectors are positioned with the broadest margin that separates the classes relative to the hyper-plane.

*2.3.5 Convolutional Neural Networks*: A Convolutional Neural Network (CNN) is a deep learning algorithm that learns from the image input using filters to capture specific features. CNNs are highly effective for image classification tasks as a type of feed-forward neural network (Qin et al., 2020). Because of the weight-sharing feature, CNNs possess high learning efficiency by decreasing the required parameters. A CNN is primarily composed of convolutional layers and pooling layers. The convolutional layers detect and extract features from the input, whereas the pooling layers condense these into denser layers for additional classification or regression operations.

## 3. Experiment

### *3.1 Data Summarization*

We used image datasets obtained from Kaggle. Specifically, we acquired the original data set of 525 bird species (Piosenka, 2023). According to the data source, Approximately 80% of the images feature males, with the remaining 20% depicting females. Males usually exhibit a wider range of colors, whereas females often have less variation in coloration. Therefore, male and female birds may not resemble each other closely. Since test and validation images primarily represent males, there might be a performance discrepancy in the classifier when it comes to recognizing female species. Due to the complexity of processing data, we divided the original data into two different datasets. The first dataset comprised images across ten distinct species, providing a manageable framework for assessing model performance on a smaller scale. The second dataset expanded this framework to one hundred species, introducing a more incredible classification difficulty. All images are in jpg format with size $224 \times 224 \times 3$ colors and were partitioned into three subsets: training set, validation set, and testing set. We used the image to train the models with labels to compare classification accuracy better. The details of the datasets are as follows in Table 1, and the preview of the dataset samples is shown in Figure 2.

**Table 1.** Details of the data used.

| Source | Data length | Data type | URL |
|---|---|---|---|
| Kaggle | 84635 train<br>2625 test<br>2625 validation | Image<br>(525 Species) | https://www.kaggle.com/datasets/gpiosenka/100-bird-species |
| Kaggle | 1607 train<br>50 test<br>50 validation | Image<br>(10 Species) | https://drive.google.com/drive/folders/19aBxP7LTd3Tul4VRkJQbrpOFPeYdOk0H?usp=sharing |
| Kaggle | 16403 train<br>500 test<br>500 validation | Image<br>(100 Species) | https://drive.google.com/drive/folders/12iudj1LmlzLydKIvvgfRY3OTAB3um-gs?usp=drive_link |


**Figure 2. Preview of the dataset samples**

### *3.2 Evaluation Criteria*

To assess the effectiveness of the classifying tasks of all models, we introduced the Accuracy, Precision, Recall, F1 Score, and Confusion Matrix as the evaluation criteria. The equations are given from (1)-(4).

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \quad (1)$$

In (1), the accuracy of the models is determined by the proportion of accurately predicted instances out of the overall number of observations.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

In (2), model precision is measured as the fraction of true positive predictions made out of all positive predictions.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

In (3), recall is calculated as the proportion of positive observations correctly predicted relative to the total in the actual class.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

In (4), the F1 Score represents the balanced mean of Precision and Recall, considering both false positives and false negatives in the calculation.

*Confusion Matrix*: A confusion matrix is a tabular representation to evaluate the efficacy of a classification model on a specific test dataset. It breaks down the predictions into four parts: true positive, false positive, true negative, and false negative.

### *3.3. Implementation of the Machine Learning Models*

*Data Pre-processing*: Use `'ImageDataGenerator'` to automatically convert image pixel values to a scale of 0 to 1.

```
datagen = ImageDataGenerator(rescale=1./255)
```

*Feature Extraction with VGG16*: Employ the pre-trained VGG16 model as a feature extractor to process images and output a processed version of the image data that captures essential features but with reduced dimensionality. Specifically, the code enters a loop for batch processing of images and features. It retrieves batches of input images and predicts their features using the VGG16 model. The current batch size is determined, and the extracted features and labels are stored accordingly. The loop continues until the specified sample count is reached or until there are no more images to process.

```
feature_extractor = VGG16(include_top=False, weights='imagenet')

def compile_features(source_dir, total_samples, samples_per_batch):
    data_gen = datagen.flow_from_directory(
        source_dir,
        target_size=(150, 150),
        batch_size=samples_per_batch,
        class_mode='binary')
    initial_data, _ = next(data_gen)
    base_features = feature_extractor.predict(initial_data)
    feature_dims = base_features.shape[1:]
    compiled_features = np.zeros((total_samples,) + feature_dims)
    compiled_labels = np.zeros((total_samples,))

    count = 0
    for data_batch, label_batch in data_gen:
        extracted_features = feature_extractor.predict(data_batch)
        batch_size = data_batch.shape[0]

        if (count + 1) * samples_per_batch > total_samples:
            extracted_features = extracted_features[:total_samples - count *
samples_per_batch]
            label_batch = label_batch[:total_samples - count *
samples_per_batch]
            batch_size = total_samples - count * samples_per_batch

        compiled_features[count * samples_per_batch : count * samples_per_batch
+ batch_size] = extracted_features
        compiled_labels[count * samples_per_batch : count * samples_per_batch +
batch_size] = label_batch

        count += 1
```

```
        if count * samples_per_batch >= total_samples:
            break

    return compiled_features, compiled_labels
```

*Label Generation*: Generate features and labels with managed batches. If the last batch exceeds the sample count, it gets trimmed to fit.

```
size_per_batch = 32
init_data_gen = datagen.flow_from_directory(
    directory_train,
    target_size=(150, 150),
    batch_size=size_per_batch,
    class_mode='binary')

indices_class = init_data_gen.class_indices

# Numerical labels to class names
map_label = dict((value, key) for key, value in
indices_class.items())

# Estimate sample count for each set
train_sample_count = *   # 1607 for class 10, 16403 for class 100
test_sample_count = *    # 50 for class 10, 500 for class 100
valid_sample_count = *   # 50 for class 10, 500 for class 100
```

*Data Reshaping for Classification*: After feature extraction, the extracted features are flattened to make them compatible with traditional classifiers. Then, Pandas DataFrames are created from the flattened feature arrays with the corresponding labels as a new column. Finally, save DataFrames as CSV files

```
train_features, train_labels = extract_features(train_dir,
train_sample_count, batch_size)
test_features, test_labels = extract_features(test_dir,
test_sample_count, batch_size)
valid_features, valid_labels = extract_features(valid_dir,
valid_sample_count, batch_size)

# Flatten the features
train_features_flatten = np.reshape(train_features,
(train_sample_count, -1))
test_features_flatten = np.reshape(test_features, (test_sample_count,
-1))
valid_features_flatten = np.reshape(valid_features,
(valid_sample_count, -1))
```

*Model Training and Evaluation*: Load the CSV files into DataFrames and split them into features and labels for training and evaluation. Three different classifiers, Logistic Regression, Random Forest, and SVM, are then initialized for performance evaluation. Metric storage dictionaries (precisions, recalls, f1_scores, error_rates) are then set up to hold various performance metrics. Inside a loop, each classifier is individually trained on the training data, used to predict labels for the validation dataset

```
# Initialize classifiers
classifiers = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(n_estimators=100),
    "Support Vector Machine": SVC()
}

# Create dictionaries to store the metrics
precisions = {}
recalls = {}
f1_scores = {}
error_rates = {}

# Train classifiers with a loop
for classifier_name, model in classifiers.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_valid)

    # Compute Evaluation Criteria
    sc_accuracy = accuracy_score(y_valid, y_pred)
    sc_precision = precision_score(y_valid, y_pred, average='macro')
    sc_recall = recall_score(y_valid, y_pred, average='macro')
    sc_f1 = f1_score(y_valid, y_pred, average='macro')
    error_rate = 1 - sc_accuracy
```

The setting of the studied machine learning models in the experiment is shown in Table 2. The Logistic Regression, Random Forest, and SVM models are trained to compare the accuracy of the species classification task for a 10-class species case and a 100-class species case.

**Table 2.** Parameter specification of the machine learning classifiers.

| Models | Details |
|---|---|
| Logistic Regression | Maximum iterations 1000 |
| Random Forest | Number of estimators 100 |
| SVM | Default parameters |

### *3.3. Implementation of the Deep Learning Model*

*Model Construction*: Define a CNN model called `'SimpleCNN'` with three convolutional layers followed by max-pooling, dropout, and fully connected layers. The convolutional layers

apply kernels to extract essential features from the input images. The spatial information is then preserved through the padding layer. The Max-pooling layers are then used to downsample the feature maps and reduce dimensionality. A dropout layer is then used to prevent overfitting. Then, two fully connected layers are connected, with the last one to produce the final classification output. The ReLU activation function is applied after each convolutional layer.

```
# Define a CNN model
class SimpleCNN(nn.Module):
    def __init__(self, num_classes=10):
        super(SimpleCNN, self).__init__()
        self.layer_conv1 = nn.Conv2d(in_channels=3, out_channels=32,
kernel_size=3, padding=1)
        self.layer_conv2 = nn.Conv2d(in_channels=32, out_channels=64,
kernel_size=3, padding=1) self.layer_conv3 =
nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
        self.layer_pool = nn.MaxPool2d(kernel_size=2, stride=2)
self.layer_dropout = nn.Dropout(0.5)
        self.layer_fc1 = nn.Linear(128 * 8 * 8, 512)
        self.layer_fc2 = nn.Linear(512, classes_count)
self.activation_relu = nn.ReLU()

    def forward(self, x):
        x =
self.layer_pool(self.activation_relu(self.layer_conv1(x)))
        x =
self.layer_pool(self.activation_relu(self.layer_conv2(x)))
        x =
self.layer_pool(self.activation_relu(self.layer_conv3(x)))

        x = x.view(x.size(0), -1)
        x = self.layer_dropout(x)
        x = self.activation_relu(self.layer_fc1(x))
        x = self.layer_fc2(x)
        return x
```

*Model Initialization*: The model is initialized in this section. The loss function is defined as the Cross-Entropy Loss, and the Adam optimizer are specified to optimize the model during training. Then, the transformations for image data are defined to prepare the data for training and testing.

*Training Loop*:

```
epochs_total = 50
for current_epoch in range(epochs_total):
    model.train()
    total_loss = 0.0
    for batch_images, batch_labels in train_loader:
```

```
        batch_images, batch_labels = batch_images.to(device),
batch_labels.to(device)

        optimizer.zero_grad()
        predictions = model(batch_images)
        batch_loss = criterion(predictions, batch_labels)
        batch_loss.backward()
        optimizer.step()

        total_loss += batch_loss.item()
```

*Validation Loop*:

```
model.eval()
    with torch.no_grad():
        correct_predictions = 0
        total_predictions = 0
        for batch_images, batch_labels in valid_loader:
            batch_images, batch_labels = batch_images.to(device),
batch_labels.to(device)
            predictions = model(batch_images)
            _, preds = torch.max(predictions.data, 1)
            total_predictions += batch_labels.size(0)
            correct_predictions += (preds ==
batch_labels).sum().item()

    history_loss_train.append(total_loss/len(train_loader))
    history_accuracy_valid.append(100 * correct_predictions /
total_predictions)

    print(f'Epoch [{current_epoch+1}/{epochs_total}], Training Loss:
{total_loss/len(train_loader)}, '
          f'Validation Accuracy: {100 * correct_predictions /
total_predictions}%')
```

*Testing Loop*:

```
model.eval()
accurate_predictions = 0
total_tests = 0
collected_labels = []
collected_predictions = []
```

```
for test_images, test_labels in test_loader:
    test_images, test_labels = test_images.to(device),
test_labels.to(device)
    predictions = model(test_images)
    _, predicted_labels = torch.max(predictions.data, 1)
    total_tests += test_labels.size(0)
    accurate_predictions += (predicted_labels ==
test_labels).sum().item()

    # Gathering labels and predictions for analysis
    collected_labels.extend(test_labels.cpu().numpy())
    collected_predictions.extend(predicted_labels.cpu().numpy())
```

The settings of the parameters of the deep learning CNN model in the experiment are shown in Table 3. All model epochs and batch size were set to 50 and 32, respectively, with adaptive moment estimation (ADAM) as the optimizer. The learning steps in the training process can be made as scale-invariant to parameter gradients with the ADAM algorithm.

**Table 3.** Parameter specification of the CNN model.

| Parameters | Value |
|---|---|
| Convolutional layer filters | 32, 64, 128 |
| Convolutional layer kernel size | 3 |
| Convolutional layer padding | 1 |
| Convolutional layer activation function | ReLU |
| Max pooling layer pool size | 2 |

## 4. Results

### *4.1. Machine Learning Models Outcomes*

The training dataset of 10 and 100 species classes are fed into the algorithms to train the classification models accordingly. These models then produce estimated class labels for bird species, which are matched against the true class labels within the test dataset. The confusion matrix comparison of the different machine learning models is shown in Figure 3. Specifically, the intensity of the blue color in these matrices corresponds to the quantity of classification within each bird species; a darker shade of blue represents a higher concentration of testing data. Moreover, an ideal classification scenario will be a dark blue concentration along the diagonal of the matrix, indicating a high number of correct predictions.

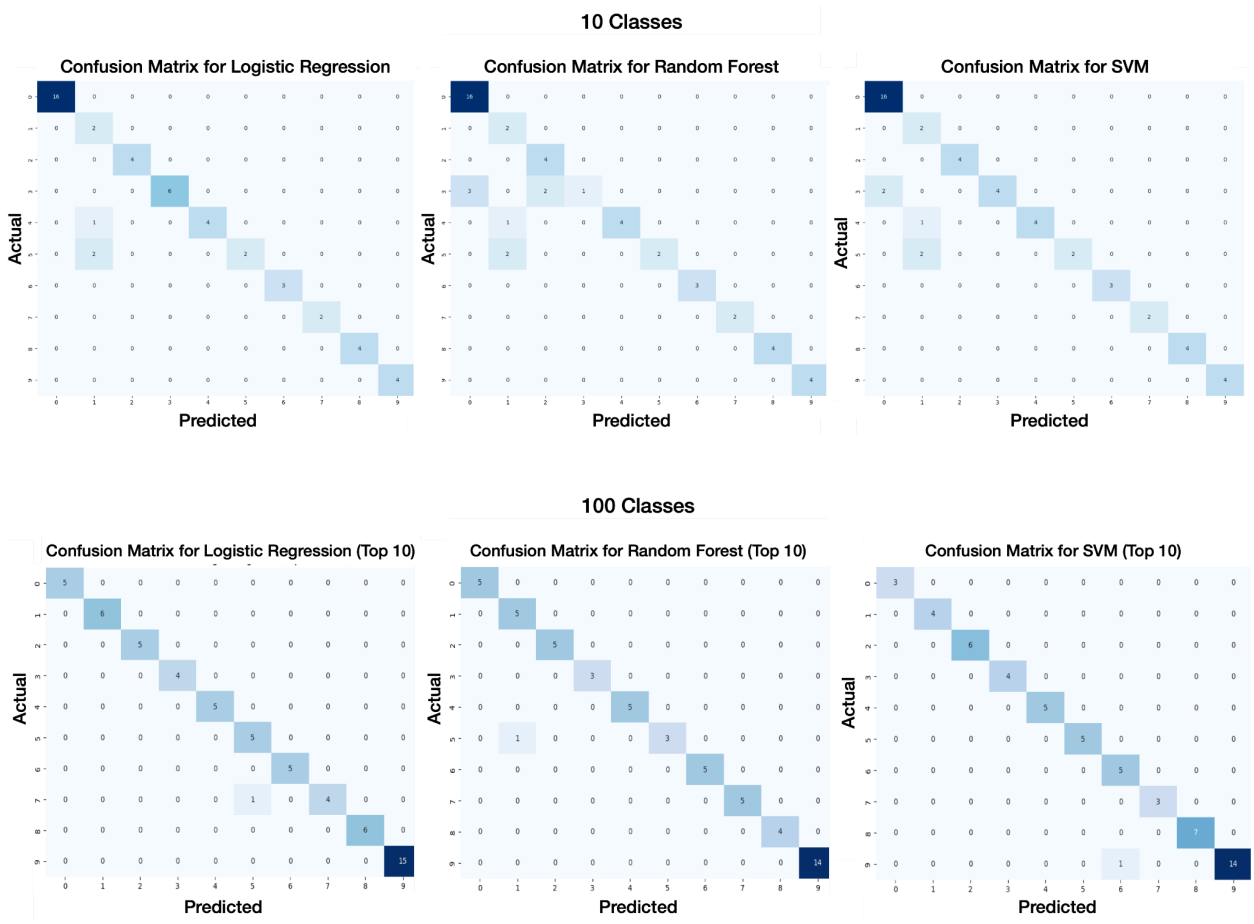


**Figure 3.** Confusion Matrices of machine learning models.

To quantify the degree of overall fitting between the actual bird species classes and bird species predicted classes, the evaluation criteria of each of the three models are calculated with different classes. The results are detailed in Table 4.

From the result in Table 3, the 10-class classification task with Logistic Regression achieves the highest accuracy at 0.9400, precision at 0.9400, and F1-score at 0.9127, with a recall slightly lower at 0.9300. The 100-class classification presents a more complex environment, and there is an overall reduction in scores across all models. Logistic Regression maintains strong performance with an accuracy of 0.8400 but shows a drop in precision to 0.8481, recall to 0.8377, and an F1-score of 0.8291. The Random Forest model experiences a notable decline with an accuracy of 0.6280, precision of 0.6551, recall of 0.6251, and F1-score of 0.6079. Among all the models, the Logistic Regression consistently shows robust performance, and the Random Forest displays potential limitations in more complex class cases.

**Table 4.** Evaluation criteria of machine learning models.

| Models | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Logistic Regression (10 classes) | 0.9400 | 0.9400 | 0.9300 | 0.9127 |
| Random Forest (10 classes) | 0.8400 | 0.8909 | 0.8467 | 0.8127 |

| Models | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| SVM (10 classes) | 0.9000 | 0.9289 | 0.8967 | 0.8868 |
| Logistic Regression (100 classes) | 0.8400 | 0.8481 | 0.8377 | 0.8291 |
| Random Forest (100 classes) | 0.6280 | 0.6551 | 0.6251 | 0.6079 |
| SVM (100 classes) | 0.7780 | 0.8099 | 0.7768 | 0.7702 |

### *4.2. Deep Learning Model Outcomes*

The result comparison of the CNN models on 10 and 100 species classes species classification of the experiment is shown in Figure 4. The CNN model is used to produce estimated class labels for bird species, which are matched against the true class labels within the test dataset. The intensity of the blue color in confusion matrices corresponds to the quantity of classification within each bird species; a darker shade of blue represents a higher concentration of testing data. Moreover, an ideal classification scenario will be a dark blue concentration along the diagonal of the matrix, indicating a high number of correctly predicted values. The loss in the training phase and validation visualization in both 10 and 100-species cases are shown in Figure 5. The x-axis denotes the number of epochs, and the y-axis denotes the training loss and accuracy, respectively. The evaluation criteria of the CNN model are calculated with different classes. The results are detailed in Table 5. For the 10-class experiment, the CNN model demonstrates a high performance with an accuracy of 0.9600, precision of 0.9633, recall of 0.9600, and an F1-score of 0.9598. The model is effective in classification and maintains a balanced precision-recall relationship.

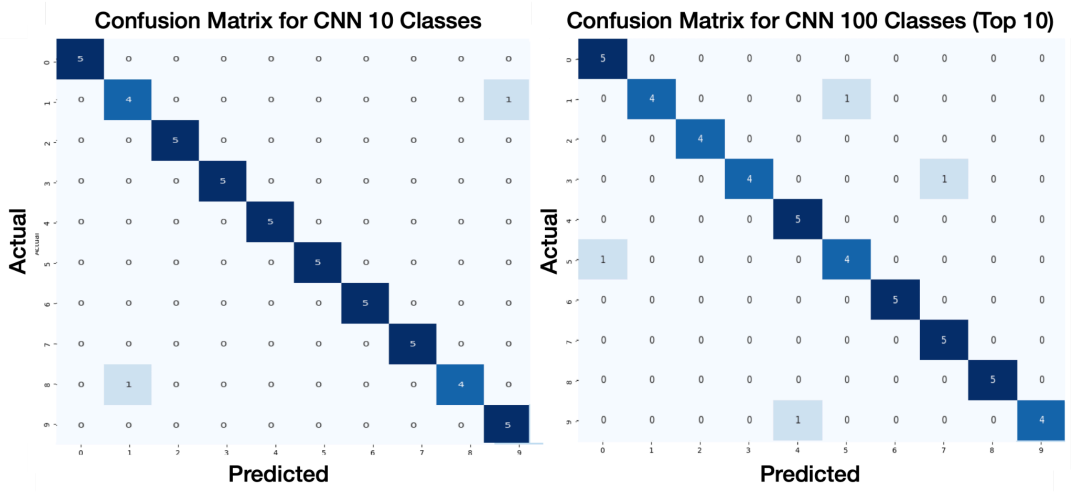


**Figure 4.** Confusion Matrices of the deep learning model.

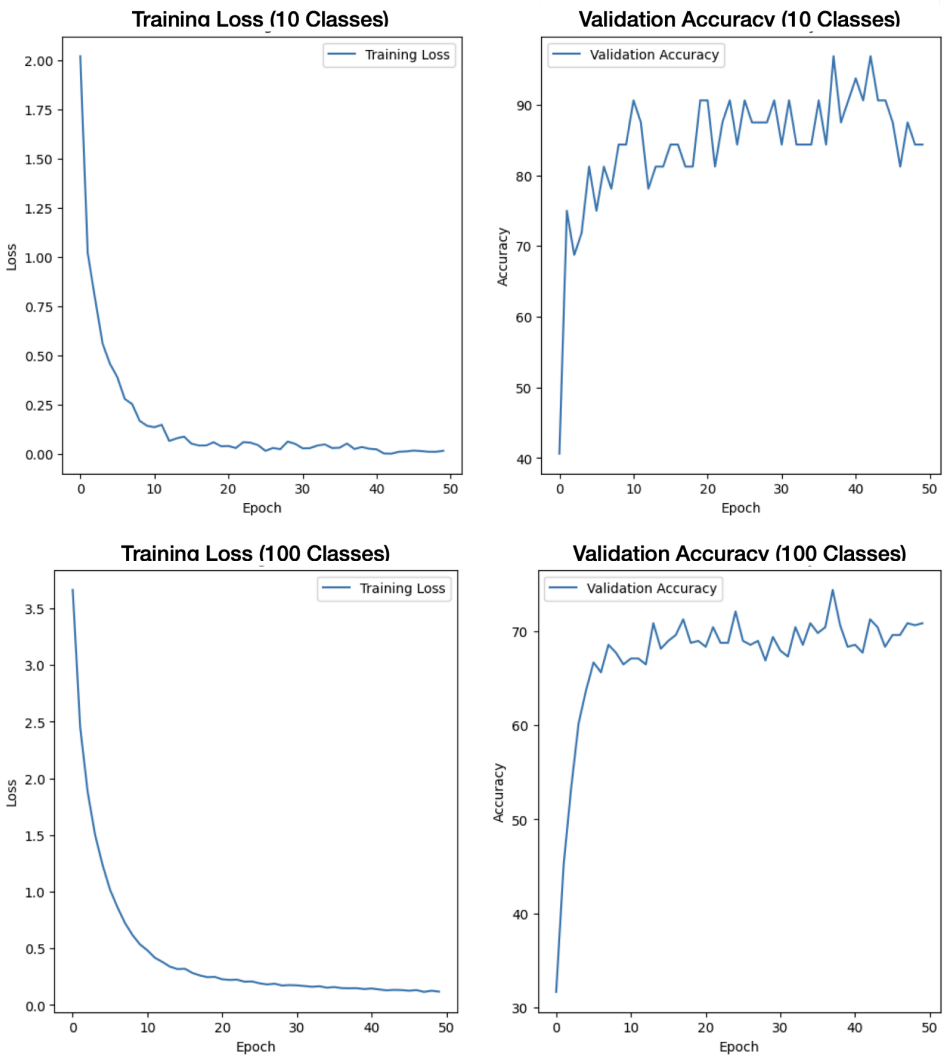


**Figure 4.** Training details of the deep learning model.

**Table 5.** Evaluation criteria of the deep learning model.

| Models | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| CNN (10 classes) | 0.9600 | 0.9633 | 0.9600 | 0.9598 |
| CNN (100 classes) | 0.7260 | 0.7470 | 0.7260 | 0.7196 |

## Discussion

This project aimed to compare traditional machine learning algorithms such as Logistic Regression, Random Forest, and SVM and a deep learning CNN for bird species classification from images. The results highlighted the strengths and limitations of each approach and underscored the challenges associated with classification tasks of varying complexity.

Logistic Regression demonstrated remarkable performance consistency across the 10-class and 100-class classification tasks. Its ability to maintain high accuracy and precision with increased class complexity is noteworthy and suggests that Logistic Regression is a reliable model for tasks of this nature. However, its performance did show signs of degradation as the complexity increased, a trend observed across all models, which indicates a potential area for optimization.

The Random Forest was robust in the 10-class task but showed a significant performance drop in the 100-class scenario. It could be attributed to the model's inherent design, which may lead to overfitting when dealing with many classes. Optimizations such as feature selection, tree pruning, or advanced ensemble techniques may enhance the scalability and performance of the method.

The SVM showed commendable adaptability. Despite not having the highest scores in any single metric, its performance was balanced. The precision and recall scores remained proportional as the complexity of the classification task increased. This balance is crucial for practical applications when the cost of false positives and negatives is significant.

The CNN excelled in the 10-class classification task, outperforming the traditional machine learning models. The architecture utilized spatial hierarchies in image data, making it particularly suited for image classification tasks. However, when scaling to 100 classes, CNN's performance decline was notable. While deep learning models are powerful, their performance in more complex classification tasks can still be improved.

Several factors could have contributed to the observed performance trends. The discrepancy in male and female bird representation in the dataset might have introduced bias, affecting the generalizability of the models. Furthermore, the choice of features, class imbalance, and hyperparameter settings could have influenced the results.

For future research, exploring techniques to mitigate class imbalance and experimenting with different feature extraction methods would be beneficial. Additionally, fine-tuning model hyperparameters, utilizing more sophisticated model architectures, or applying data augmentation strategies could enhance the models' ability to handle more classes more effectively. The integration of transfer learning, where a model is pre-trained on a large dataset and tuned on a specific task, could also be explored to improve model performance.

Moreover, considering the dynamic nature of ecological environments, future studies could focus on real-time classification systems capable of adapting to new data and evolving classification requirements. Implementing incremental learning systems, where the model updates its knowledge without retraining from scratch, might be a valuable addition to the field of ecological monitoring and species classification.

In summary, this project provides valuable insights into the capabilities and limitations of different machine learning and deep learning models for bird species classification. It sets the stage for further research aimed at improving model accuracy, efficiency, and scalability to face more complex classification tasks in ecology.

## References

Christin, S., Hervet, É., & Lecomte, N. (2019). Applications for deep learning in ecology. *Methods in Ecology and Evolution*, 10(10), 1632–1644. https://doi.org/10.1111/2041-210X.13256

Depauw, L., Blondeel, H., De Lombaerde, E., De Pauw, K., Landuyt, D., Lorer, E., Vangansbeke, P., Vanneste, T., Verheyen, K., & De Frenne, P. (2022). The use of photos to investigate ecological change. *Journal of Ecology*, 110, 1220– 1236. https://doi.org/10.1111/1365-2745.13876

Elhamod, M., Diamond, K. M., Maga, A. M., Bakis, Y., Bart Jr, H. L., Mabee, P., ... & Karpatne, A. (2022). Hierarchy‐guided neural network for species classification. *Methods in Ecology and Evolution*, 13(3), 642-652. https://doi.org/10.1111/2041-210X.13768

Gerry. (2023). Bird 525 Species - Image Classification. Kaggle.

Nazish, Ullah, S. I., Salam, A., Ullah, W., & Imad, M. (2021). COVID-19 lung image classification based on logistic regression and support vector machine. In *European, Asian, Middle Eastern, North African Conference on Management & Information Systems* (pp. 13-23). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-77246-8_2

Piosenka, G. (2023). 525 Bird Species. Kaggle. Retrieved November 10, 2023, from https://www.kaggle.com/datasets/gpiosenka/100-bird-species

Qin, J., Pan, W., Xiang, X., Tan, Y., & Hou, G. (2020). A biological image classification method based on improved CNN. *Ecological Informatics*, *58*, 101093. https://doi.org/10.1016/j.ecoinf.2020.101093

Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN COMPUT. SCI*. 2, 420 (2021). https://doi.org/10.1007/s42979-021-00815-1

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv*:1409.1556. https://doi.org/10.48550/arXiv.1409.1556

Tammina, S. (2019). Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)*, *9*(10), 143-150. http://dx.doi.org/10.29322/IJSRP.9.10.2019.p9420