REST Journal on Data Analytics and Artificial Intelligence

Vol: 4(4), December 2025

REST Publisher; ISSN: 2583-5564

Website: http://restpublisher.com/journals/jdaai/

DOI: https://doi.org/10.46632/jdaai/4/4/8

Birds Species Identification Using Deep Learning

C. Prakash Narayanan, A.R. Ashok Kumar, K. Nandhini, *A. Aiesha

P.S.V College of Engineering and Technology, Krishnagiri, Tamil Nadu, India.

*Corresponding Author Email: *ansaraiesha0220@gmail.com*

***Abstract**: Classifying bird species presents significant challenges and often leads to ambiguous identifications. Even expert ornithologists may have differing opinions when examining bird images. This challenge tests the limits of visual identification capabilities, both for humans and computers. Although birds share common basic body structures, their shapes and appearances can vary considerably among different species. Significant variations are also observed even within the same species, due to differences in lighting conditions, backgrounds, and especially varying postures. Examples of these postures include birds in flight, aquatic birds, and birds perched partially concealed by foliage. This project aims to utilize machine learning capabilities to help birdwatching enthusiasts identify bird species based on the photographs they take.*

***Keyword**: Data preparation, multilayer perceptron, deep convolutional networks, machine learning algorithms.*

1. INTRODUCTION

Bird behavior patterns and population changes have emerged as contemporary concerns. Birds act as indicators for detecting other organisms within ecosystems. A key challenge in ecology, the science that studies how organisms interact with their environment, involves monitoring bird population dynamics. Monitoring and classifying birds in their natural habitats through sound has recently attracted considerable attention. Image-based species classification proves useful for applications such as monitoring breeding patterns, assessing biodiversity, and studying population changes. Birdwatching has become a hobby that offers a respite from daily routines. At the same time, since birds are essential to our ecosystems, it also represents our responsibility to understand the natural world. Those who engage in this activity are called bird enthusiasts or birdwatchers. Ornithology is the official scientific discipline dedicated to the study of birds, and professional researchers who specialize in birds are called ornithologists. According to recent research conducted by the American Museum of Natural History, there are approximately 18,000 bird species on Earth. Even though some birds may appear similar or are believed to interbreed, they belong to different species. However, due to practical limitations such as geographical location, viewing distance, and available equipment, directly identifying birds relies on basic characteristics, and accurately identifying them using distinctive features is often considered a challenging classification process. While birds can be manually classified by domain experts, this approach becomes increasingly time-consuming as the volume of data increases. Subsequently, due to complex variations and edge cases, identifying the components of the object becomes a significant challenge. The primary objective is to develop bird identification technology that creates a digital repository of bird species—essentially building a gallery for future generations. While our ancestors recorded historical information in books and documents, we need to preserve data for future generations through technological means. Therefore, this technology was developed to allow anyone to easily verify information about bird species through image recognition. This research developed an approach that uses convolutional neural networks to analyze the visual characteristics of birds in their natural habitats by identifying localized features from archived or live bird images. First, raw input data containing various anatomical parts of the birds was collected and segmented. Subsequently, feature vectors for individual body parts are identified, collected, and refined based on their shape, size, and color characteristics. Finally, a CNN model was trained using bird images to extract features, taking into account specific attributes and predetermined characteristics; the classified and trained data was then stored on a server for identifying the target object.

Implementation: Following this detailed explanation, we can implement our CNN. To distinguish cat images from dog images, we will be using Kaggle's 'Dogs vs. Cats' dataset. Our architecture will consist of four convolution and pooling layer combinations, followed by two fully connected layers. This network will take a cat or dog image as input and provide a binary output.

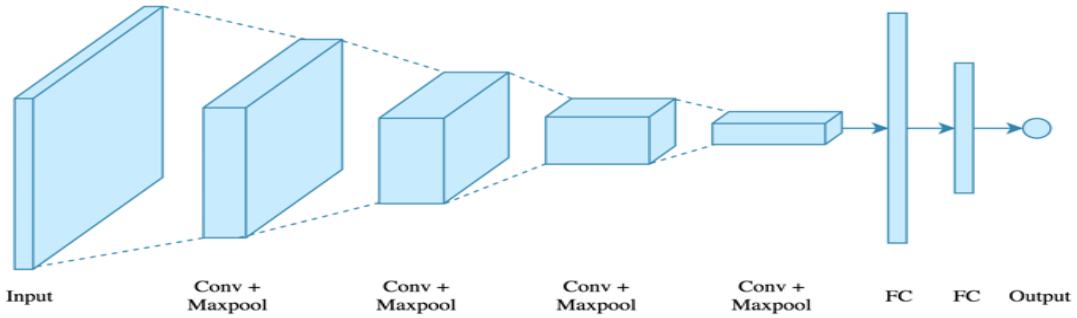

FIGURE 1.

2. DESIGN

A. File Design: The file system is a crucial aspect that significantly impacts computer performance. File formatting deals with two main components: file size and data redundancy. At the same time, all files are structured to include detailed information associated with each block. A monolithic database containing information for all components increases the complexity of the system. In contrast, a relational database comprises multiple tables, along with mechanisms to facilitate interaction between them. Data relationships between the tables can be linked, integrated, and presented through database forms. Most relational database systems provide capabilities for data sharing.

- Through network connections
- Via internet protocols
- Using portable electronic devices such as mobile devices and PDAs
- With additional software platforms

Therefore, a relational database system is used in the bird species identification system. The basic design files are crucial for the functioning of this system. The system's performance depends on the quality of its design. Considerable attention has been paid to reducing the size of the files and eliminating unnecessary redundancies. At the same time, all files are structured to include complete information about each company. A single integrated database containing all company information would unnecessarily complicate the system. Individual database tables will be linked together; this will be explained in subsequent chapters on table structure and the normalization process.

B. Input Design: Input design refers to the process of obtaining accurate information from users. This verified information is then stored as processing data in the database. The purpose of the input format is to enable straightforward and accurate data entry. Input screens ensure that incorrect data is filtered out, preventing it from becoming operational data during the input process. Input design is a part of the overall system design process; it requires careful consideration and also constitutes a costly development phase. Since it acts as the primary interface between the users and the computer system, it is highly susceptible to errors. When incorrect data enters the system, the processing and output functions can magnify these errors. The objectives of input design are described below:

Developing economical input methods

- Achieving high levels of accuracy
- Eliminating ambiguity

The various stages of the input process include:

- Data recording
- Data validation
- Data transfer to the system
- Data correction

Input design refers to converting raw inputs into a format suitable for the computer system. The purpose of input design is to simplify data entry and prevent logical errors. The computer system processes the following types of input data:

i. External Inputs: These represent the key inputs for the computer system. External input refers to the data provided by the users. The system handles two types of external inputs: one coming from regular users, and the other from administrators.

ii. Internal Inputs: These are the input formats required for the computer system to understand. Once external input is received from the users, this data is sent as messages through the windowing system. These messages are then received and processed as input for further operations.

3. METHODOLOGY

A. Dataset: Deep learning operates in a way similar to how the human brain works. It learns from data and uses training information to make inferences about the characteristics of the data. As a result, a diverse and extensive dataset is required to create an effective neural network model. For this reason, our research utilizes data augmentation techniques. These techniques increase the number of training examples for each category, mitigate class imbalance problems, and allow the neural network model to learn from diverse datasets, appropriate image augmentation techniques are selected. These techniques include Gaussian noise addition, Gaussian blur application, image flipping, contrast adjustment, hue shifting, additive operations (adding values to pixel channels), multiplicative operations (multiplying pixel channel values), sharpening, and affine transformations. A comprehensive dataset also helps prevent overfitting problems that frequently occur in deep network training. Compared to text-based datasets, image datasets require more computational resources. Therefore, the objective of our research is to reduce these computational requirements by removing unnecessary parts from the images. This also reduces the number of pixels that the neural network model needs to process. The pre-trained object detection deep neural networks remove background elements or regions, isolating features only from the bodies of the birds. This model uses Mask R-CNN to detect birds in each image throughout the training and inference phases. We used pre-trained Mask R-CNN parameters developed using the COCO dataset; this dataset includes 1.5 million object instances spread across 80 object categories, including birds. Testing refers to running a program to find errors. It is a crucial component of software quality assurance and also serves as a final review of the specifications, design, and code. Software testing is a critical phase and presents an interesting challenge in software development. Therefore, before user acceptance testing, several testing procedures are conducted on the proposed system. An effective testing process has a higher chance of detecting a previously undiscovered error. A successful test reveals a previously unidentified defect.

B. Output Design: The output design specifies the required output and its presentation format. Great care must be taken to provide accurate information to facilitate sound decision-making.

The generated output is divided into three main categories:

- Output displayed on the screen
- Output stored as files on storage devices
- Printed physical output

Screen output mainly shows the produced results on a display monitor. Query results are generally displayed on the screen for immediate online information access. The ability to save the generated output to files enables future reference and hard copy production as needed for management reporting and situational requirements.

Objectives of Output Design:

- Design the output to fulfill its purpose
- Design the output to suit user needs
- Provide appropriate output volumes
- Ensure the output reaches its intended destination
- Deliver the output on time
- Select the appropriate output method.

C. Data Dictionary: Following user requirement analysis, all data storage needs are structured into tables. These tables undergo normalization to avoid inconsistencies during data entry processes. A data dictionary consists of a file or collection of files that hold a database's metadata. It includes information about other database objects, such as data ownership, object relationships, and supplementary data. The data dictionary serves as a crucial element in relational databases.

What is Data Dictionary?

A data dictionary holds metadata, which is information describing the database. This is important because it maintains records of database content, user authorization levels, and where data is physically stored. Regular database users don't typically access the data dictionary; database administrators exclusively manage it.

A data dictionary generally includes information regarding:

- The names and structures of all tables in the database Table specifications, such as ownership, access permissions, and when they were created
- Physical storage information about the tables, including location and storage methods
- Table constraints, such as primary and foreign key attributes
- Information about defined database views

- Analysts use data dictionaries for these purposes:
- Managing details in large-scale systems
- Establishing common definitions for all components
- Documenting system features
- Identifying system errors and inconsistencies

D. Logical Database Design: A data dictionary comprises one or more files containing database metadata. It maintains records about other database objects, such as ownership details, inter-object relationships, and supplementary information. The data dictionary represents an essential element in relational databases. Ironically, while it's critically important, most database users cannot see it. Generally, only database administrators have the ability to view and manipulate the data dictionary.

E. Database Design: Database design involves organizing database files that function as the main information repositories for a computer system. These files require careful planning and design to gather, arrange, and update required information. Among the goals of database design are delivering effective secondary storage and enhancing the overall performance of the computer system's program elements.

F. What is a dataset?

In machine learning, a dataset refers to the data provided to an algorithm to train a model. A dataset consists of interconnected, distinct data elements; these can be retrieved individually, collectively, or as a whole. It follows a specific organizational structure. For example, in a database, a dataset might include business-related information such as employee names, salaries, contact information, and revenue figures. A database can be considered entirely as a dataset, and so can portions of the data that focus on specific types of information, such as departmental sales figures. A dataset typically consists of a collection of data, such as a single database table or a statistical matrix; in which the columns represent different variables, and the rows correspond to the individual members within the dataset. This dataset holds the values of every variable for each member (such as an object's height and weight). Individual values are referred to as data points. A dataset can include data for single or multiple members; each linked to a sequence number. The term dataset can also be applied more generally to describe data within a group of interrelated tables connected to a specific experiment or occurrence. In machine learning, datasets denote the data fed into algorithms for creating models. A dataset comprises interconnected, separate data elements that can be retrieved one at a time, as a group, or handled as a single unit arranged in a particular data structure. For instance, within a database context, a dataset may hold corporate information including employee names, compensation, contact information, and revenue data. Both the entire database and particular data subsets within it—like sales data by department—qualify as datasets. A dataset is an assembly of data that usually corresponds to the content of one database table or a statistical matrix. In this format, columns denote specific variables, and rows signify individual dataset members. each separate value is termed a data point. A dataset may hold data for one or several members, indicated by the row count. The term can also apply more generally to data across interrelated tables linked to a specific experiment or occurrence. An illustration of this would be datasets gathered by space agencies performing experiments using spacecraft instruments. Datasets too massive to be effectively managed by traditional data processing tools are known as big data.

4. TYPES OF TESTS

A. Unit Testing: Unit testing involves verifying that individual code components work correctly, typically focusing on single functions. When working with object-oriented programming, these tests usually target classes and their fundamental components such as constructors and destructors. Developers typically create unit tests during coding (using a white-box approach) to ensure that their functions behave as intended. A single function may require multiple tests to handle edge cases and different execution paths. While unit testing cannot guarantee that an entire software system functions correctly, it verifies that individual components work correctly in isolation. This development practice integrates various strategies for preventing and detecting defects, helping to mitigate project risks, timelines, and costs. Developers and engineers conduct these tests throughout the building phase of the development process. The objective is to discover and resolve coding problems at an early stage, prior to advancing the code to broader testing phases. This strategy aims to enhance both the end product's quality and the development process's overall productivity.

B. Integration Testing: Integration testing examines how different software components interact with each other and verifies that their interfaces conform to the design specifications. The components can be integrated gradually or simultaneously; however, the

gradual approach is generally preferred as it simplifies the identification and resolution of interface problems. This testing phase focuses on finding defects at the points where integrated modules connect and interact. The testing progresses by gradually combining larger sets of verified components that are relevant to the computer system; this continues until the entire software functions as a unified system.

C. Functional Testing: Functional testing involves verifying whether specific actions or features in the code perform as intended. These tests are typically based on requirements documents, although some development approaches derive them from use cases or user stories. The main questions addressed by functional tests are whether users can perform certain tasks and whether specific features function correctly. This testing method delivers structured proof that the functionalities being tested operate in accordance with business needs, technical requirements, system documentation, and user manuals.

Functional testing concentrates on these elements:
Valid Input: Acceptable data categories must be correctly handled.
Invalid Input: Unacceptable data categories must be appropriately denied.
Features: The specified functionalities must be implemented and tested.
Results: The designated types of application outputs must be generated and verified.
External Systems/Workflows: The connected systems or processes must be implemented.

System Testing: This testing phase evaluates whether the fully integrated system meets its requirements. For example, a system test might involve verifying a login interface, followed by creating and editing a record, then generating or printing reports, performing summary functions or deleting (or archiving) records, and concluding with logging out.
Performance testing: This type of testing evaluates how quickly and efficiently a system operates and ensures that it delivers results within the deadlines defined in the performance specifications. It belongs to the category of black-box testing methods.

D. White-Box Testing: Within software testing, the box testing methodology encompasses black-box and white-box techniques. White-box testing, alternatively called glass-box, structural, clear-box, open-box, or transparent-box testing, analyzes the software's internal code and architecture by confirming that particular inputs generate the anticipated outputs. This approach focuses on testing the internal workings of an application and its internal structure. Conducting this type of testing requires programming expertise to create appropriate test cases. The main objective of white-box testing is to examine how data travels through the software and to improve its security. The term 'white box' comes from viewing the system from an internal perspective. Terms like clear box, white box, or transparent box indicate the capability to look past the software's outer layer into its inner operations. Test cases for this method are developed during the design stage of the software development lifecycle. White-box testing employs multiple techniques including data flow analysis, control flow analysis, path testing, branch testing, and statement and decision coverage as guidelines for building error-free software.

E. White-Box Testing Techniques

- Testing data flow
- Testing control flow
- Testing branch coverage
- Testing statement coverage
- Testing decision coverage

Data Flow Testing: Data flow testing analyzes how data moves through a program by monitoring how variables carry information throughout the code. Its objective is to collect specific details about the data at each stage of the program execution process. This testing method employs different techniques to examine the control flow of the program; it examines how variables change according to the sequence of events. It focuses on two main areas: where values are assigned to variables and where those values are actually used. By focusing on these locations, the flow of data can be effectively tested.

Control Flow Testing: Control flow testing is a white-box testing technique that seeks to determine the execution order of program statements and instructions by examining the control structure. Test cases are created based on how the program's control structure is organized. With this technique, testers select specific sections of larger programs to establish test paths. This method is frequently used during unit testing. The program's control graph represents the test cases, and a control flow graph is constructed using nodes, edges, decision nodes, and junction nodes to map all possible execution paths.

Branch Coverage Testing: Branch coverage is an approach that guarantees every branch within a control flow graph undergoes testing. It confirms that each potential result (true and false) at every decision point is run at least one time. As a white-box testing method, branch coverage ensures that every branch originating from each decision point is executed. Statement Coverage Testing: Statement coverage testing is a widely employed software testing approach within white-box testing. This method is applied during white-box test case creation and mandates that each statement in the source code runs at least once. It calculates the proportion of executed statements relative to all statements present in the source code. Statement coverage testing produces test case scenarios in the white-box testing framework according to the code's architecture.

Decision Coverage Testing: Decision coverage is a white-box testing method that assesses Boolean value results. It tracks both the true and false outcomes generated by Boolean expressions. When statements (control flow statements) such as do-while loops, if statements, and case statements can produce multiple outcomes, they are considered decision points because they yield either true or false results. This technique, using a control flow graph or diagram, ensures that all possible outcomes of every Boolean condition in the code are tested.

Grey box Testing: Gray box testing is a software testing technique that evaluates applications with partial understanding of their internal architecture. It merges aspects of black-box and white-box testing by utilizing internal code access for test case development (like white-box testing) while performing tests at the functionality level. This method is particularly useful in identifying context-specific issues in web-based systems. For example, if a tester finds a defect, they can modify the code to fix it and immediately retest the application. Gray box testing extends test coverage to all layers of complex software systems, allowing testers to evaluate both the user interface and the underlying code structure. It is primarily used in integration testing and penetration testing scenarios

5. SYSTEM STUDY

A. Existing System: This employs the CNN (VGG16) algorithm for bird species identification from images. The method utilizes unprocessed data without preprocessing, and because the VGG16 model contains a limited number of layers, it reaches an accuracy ceiling of around 80%.

Drawbacks

- The raw data is used without pre-processing.
- This system can classify only 200 bird species.
- Low accuracy rates (78% to 80%)

B. Proposed System: The substantial volumes of data produced for individual bird identification are too intricate and expansive to manage with traditional methods. Data mining supplies the essential tools and methodologies to convert these massive data quantities into useful information for making decisions. Employing data mining techniques enhances accuracy while decreasing processing time. This project examines various research papers that utilize one or more data mining algorithms for predicting heart disease. In one paper, the results obtained from neural network applications achieved nearly 100% accuracy. Consequently, predictions generated through data mining algorithms produce effective outcomes. Utilizing data mining methods on heart disease treatment data can achieve dependable performance similar to diagnostic outcomes. Our system employs the CNN (Efficient Net B3) algorithm for bird species classification from images, and prior to introducing the dataset into the program, we implement a preprocessing technique known as Image Data Generator to boost the program's accuracy.

Advantages

- Data mining techniques improve accuracy and reduce computation time.
- This project reviews various research papers that use one or more data mining algorithms for identifying bird species.
- The neural network results in one paper approached nearly 100%, proving that data mining algorithms yield efficient prediction results.
- This system can classify up to 325 bird species.
- Its accuracy exceeds 90%.

6. CONCLUSIONS

This project classifies bird species using deep learning methods for image recognition. The CNN (Convolutional Neural Network) algorithm serves as an effective solution for classification and attaining high accuracy levels. The created system connects to a user-friendly website where users can submit bird photographs and receive identification results. With this project, we can readily determine a bird's species from photographed images. Our algorithm delivers an accuracy rate of 95.09%.

Future Work: By training the computer with additional data, it can identify more items. Expanding the training dataset improves prediction accuracy. Developing a mobile application for Android or iOS will offer greater user convenience than a website. This

system can be deployed on cloud infrastructure, which provides extensive data storage capabilities for comparisons and substantial computing resources for processing tasks (especially for neural networks).

REFERENCES

[1]. Tayal, Madhuri, Atharva Mangrulkar, Purvashree Waldey, and Chitra Dangra. 2018. "Bird Identification by Image Recognition." Helix 8(6): 4349–4352.

[2]. Albustanji, Abeer. 2019. "Veiled-Face Recognition Using Deep Learning." Mutah University.

[3]. Alter, Anne L, and Karen M Wang. 2017. "An Exploration of Computer Vision Techniques for Bird Species Classification.".

[4]. Atanbori, John et al. 2018. "Classification of Bird Species from Video Using Appearance and Motion Features" Ecological Informatics 48: 12– 23.

[5]. Brownlee, Jason. 2016. "How To Use Classification Machine Learning Algorithms in Weka." Retrieved from https://machinelearningmastery. com/useclassification-machine-learning-algorithmsweka/. [6] Cai, J., Ee, D., Pham, B., Roe, P., & Zhang, J. (2007, December). Sensor network for the monitoring of ecosystem: Bird species recognition. In 2007 3rd international conference on intelligent sensors, sensor networks and information 293-298. IEEE.