Bird Species Recognition using Hybrid ML

Rahul Kulkarni, Prachi , Sai Vaibhav Medavarapu, Shaik Shahid Ali, Sidde Devesh,
Gujjula Yaswanth Kumar Reddy,Kunnam Varun Reddy, Bhuma Yaswanth Reddy

Abstract

In several disciplines, including ornithology, ecology, and conservation, identifying the different species of birds is essential. Traditional techniques that rely on eye observation or audio recordings have accuracy and efficacy restrictions. In this abstract, we provide a multimodal method for identifying different bird species that incorporates auditory and picture analysis. Our method attempts to offer a more complete representation of bird species, including both visual and aural properties, by extracting features from bird photos and audio recordings and combining them at the feature or decision level. The multimodal technique has potential applications in bird monitoring, biodiversity assessment, and cit- izen science and outperforms single-modal approaches according to experiments using real-world datasets. When compared to conventional techniques, the suggested multi- modal approach has the potential to greatly enhance bird species identification. Our method seeks to improve the precision and effectiveness of bird identification by utilising the power of deep learning and merging data from both image and auditory modalities. Particularly in situations when visual signals or vocalisations alone may not be sufficient, the combination of visual and auditory traits can offer a more robust and consistent iden- tification of bird species. This multimodal technique has potential implications in many areas, advancing citizen science projects, conservation efforts, and avian study. For bird enthusiasts, researchers, and conservationists alike, further study and improvement in this field may produce useful tools.

Chapter 1 INTRODUCTION

1.1 Introduction

An increasingly prominent area of research in ornithology and birdwatching is the iden- tification of bird species using image and audio analysis. Using high-resolution cameras, audio recorders, and machine learning algorithms, it is now possible to positively identify different bird species based on their distinctive visual and audible traits. Photographing or filming birds in their natural environments allows for the visual identification of bird species utilising photos. Utilising a variety of methods, including computer vision and picture recognition algorithms, it is possible to identify important details in these pho- tographs, such as colour patterns, body shapes, bill shapes, and other visual cues that are distinctive to specific bird species. Large datasets of labelled bird image collections can be used to train machine learning algorithms to spot trends and make precise identi- fications. Similar to visual identification, audio identification of birds requires recording their vocalisations and songs with audio recorders. To detect distinctive vocal patterns connected to various bird species, these recordings can be examined using spectrograms, which are visual representations of the frequency and amplitude of sound waves. Large datasets of labelled bird vocalisations can be used to train machine learning algorithms to recognise certain vocal patterns and make precise identifications. Combining image and audio analysis can offer a more thorough and trustworthy method of identifying different kinds of birds. For instance, audio analysis can be useful for identifying birds in situa- tions where they may be visually concealed by foliage or other environmental conditions. Additionally, it's possible for some bird species to exhibit identical visual traits but dif- ferent vocalisations, or vice versa. For this reason, combining image and audio analysis is crucial for precise identification. Overall, the use of image and audio analysis in bird species identification presents a promising strategy for increasing our knowledge of the diversity, distribution, and behaviour of birds. It also has the potential to support citizen science projects and conservation efforts aimed at bird monitoring andconservation.

1.1.1 Transfer Learning

The use of transfer learning in classroom exercises is common. The more factors that two individuals share, the easier it is to transfer learning; otherwise, ”negative transfer”—or increased difficulty—occurs. For instance, due to the two different models of the centre of gravity position, learning to ride a bicycle does not adapt to learning to ride a tricycle. The questions of ”what to transfer,” ”how to transfer,” and ”when to transfer” are the three primary issues that need to be researched in transfer learning. What can be moved between domains or tasks is what the phrase ”what to transfer” alludes to. This knowledge can be divided into two categories: specific knowledge, which is knowledge that is unique to a task or field; and general knowledge. The target locality energy can be improved by attempting to find those common knowledge components that are shared between the source field and the target field. Transfer learning techniques can be categorised into the four following groups, as per the study question ”what to transfer”: instance-based transfer learning, feature-based transfer learning, parameter-based transfer learning,and relation-based transfer learning.

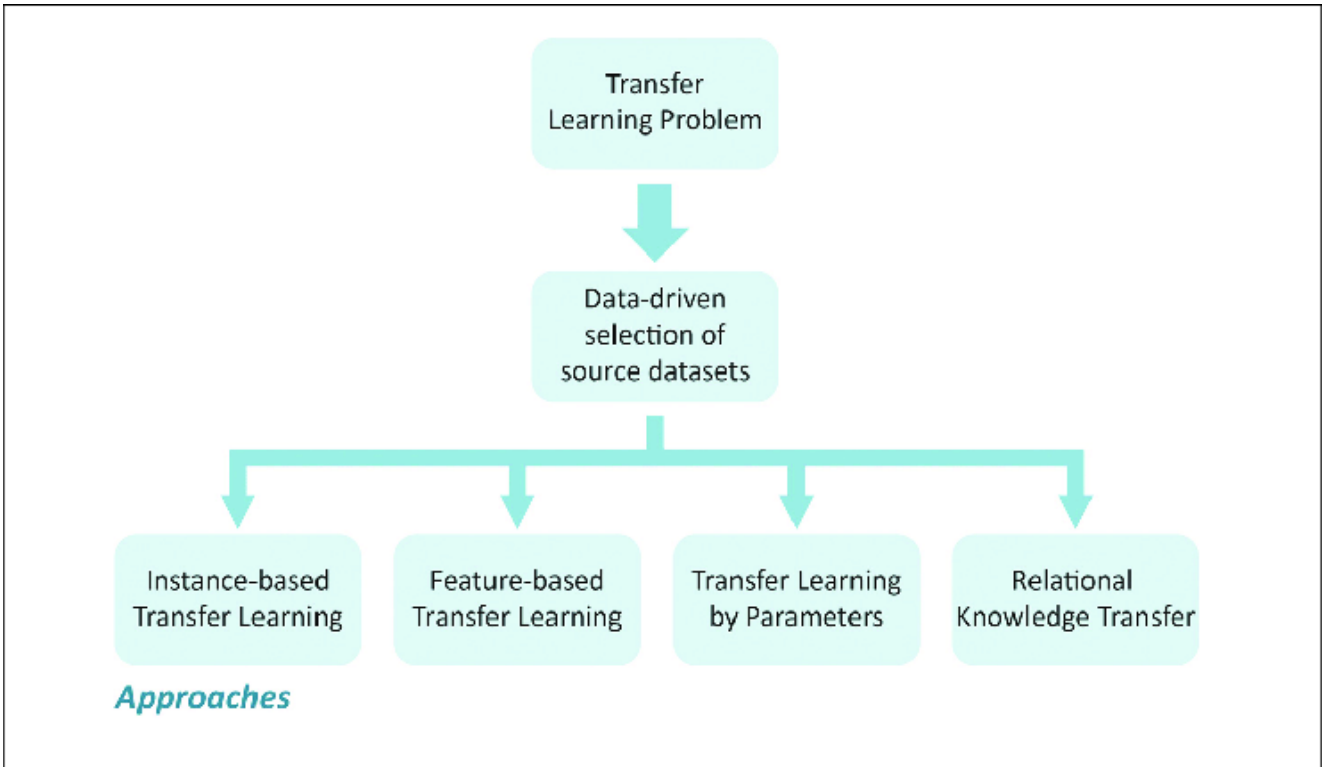

Figure 1.1: Transfer Learning

1.1.2 Instance-based transfer learning method

The primary tenet of the instance-based transfer learning approach is that the weight of the source domain sample will change depending on how significant a role it plays in the training of the target domain model. If not, the source domain sample's weight will be decreased. Maximising the contribution of knowledge from the source field to learning in the target field is the ultimate objective. Positive transfer learning is a challenge that can be effectively overcome using an iterative learning approach called TrAdaBoost, which was first proposed in. The upper bound of the generalisation error in the model was determined based on the theory of probability approximately correct (PAC). On the one hand, the weight of "bad" data was lowered, on the other hand, the weight of "good" data was increased. This commonly used transfer learning method based on the fundamentals of boosting has some drawbacks, such as weight mismatch, disregard for the first half of the classifier, imbalance in sample size, and too rapid decline in source domain weight. For the purpose of resolving the issue of transfer learning in natural language processing scenarios, other research developed an instance-weight architecture. This type of method is typically applicable to scenarios with small differences between source field distribution and target field distribution and does not apply to some complex computer vision tasks, despite the fact that the weight method based on instances is not difficult to obtain the generalisation error's upper bound and has strong theoretical support.

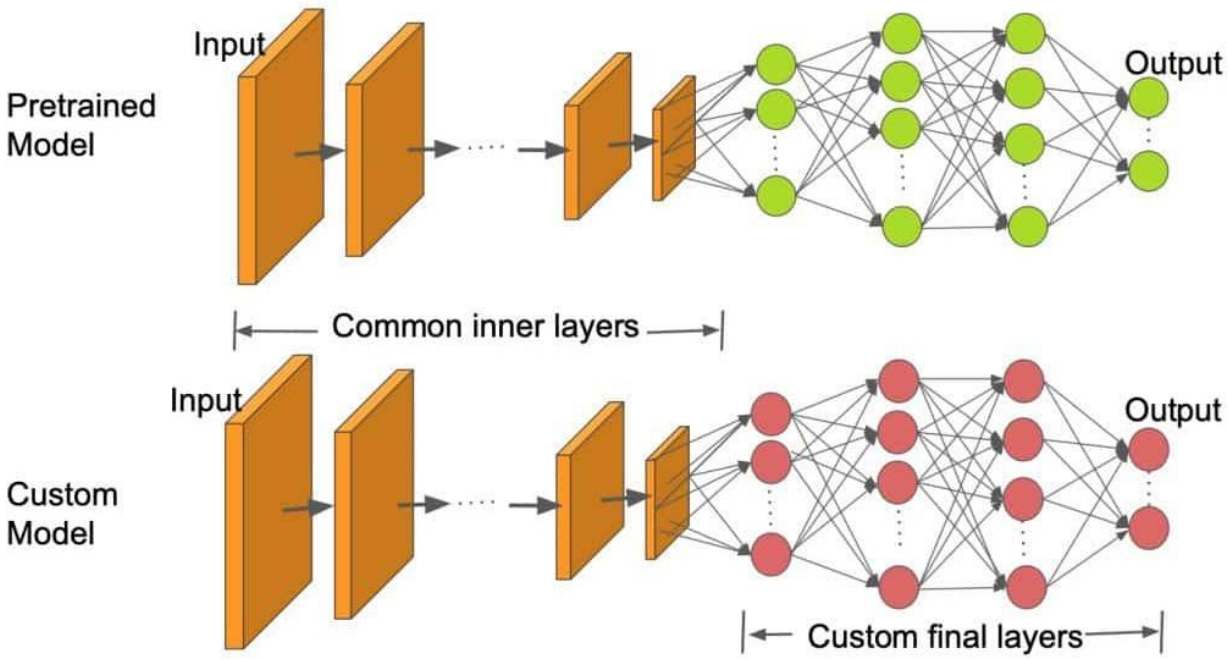

Figure 1.2: Instance-based transfer learning method

1.1.3 Feature-based transfer learning

The main goal of feature-based transfer learning is to reduce the distribution gap between the source field and the target field in the new subspace, even if domain-sharing features are strengthened while exclusive features are weakened. This is done by directly using source domain data samples for training after obtaining the new feature representation of samples in the new space through feature transformation technologylearning. Compara- tively speaking, this strategy is more transferable than the instance weight method. The two categories of feature-based approaches are feature extraction and feature selection. The maximum mean discrepancy embedding (MMDE) method was suggested by Pan et al. Its goal is to reduce the original feature discrepancies between domains in the new feature space by mapping the data features between domains to a new space. Transfer component analysis (TCA) algorithm was introduced to lessen the computational strain due to the high computational complexity of MMDE. A sparse feature coding representa- tion that primarily works with the source domain data and produces a feature dictionary was proposed by Raina et al. The base vector's matrix, which makes up the dictio- nary's characteristics, contains the crucial aspect of the data. Data between domains are expressed via basis vector coding, provided that the base vectors' properties may signif- icantly lessen discrepancies between domains. The way of thinking, which is based on maximum average differences and regular risk reduction, is known as a maximum interval straight push migration method of study. The authors attempted to identify a new feature representation in the feature mapping space between the source domain and the target domain under the framework of classification regularisation, then attempt to lessen the gap between the domains, and ultimately achieve knowledge transfer.

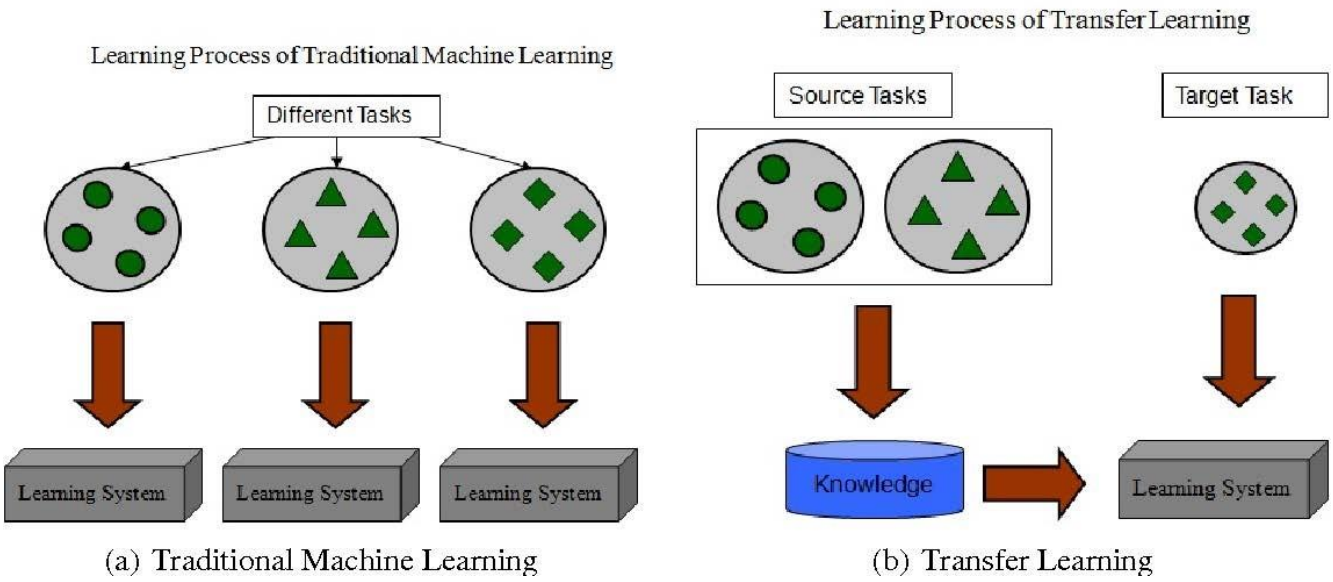

Figure 1.3: Feature-based transfer learning

1.1.4 Transfer learning method based on the structural relation- ship

The most challenging learning scenario in the transfer learning approach is the one based on the structure relation. mostly due to the fact that the source domain sample and the target domain sample in the instance do not share any characteristics. To lessen the differences between domains, however, it is challenging to map the features into the new subspace using feature transformation. Only the structural relationship between the data, such as their proximity to one another, can be used to transmit relevant knowledge across the source and target domains. In order to improve the performance of the model, the transfer learning approach based on the structural connection uses the structural relationship as the knowledge of the transfer and shares the structural relationship between the source domain and the target domain.

1.1.5 Pre-processing

When employing picture and audio analysis to identify different bird species, pre-processing is a crucial step. Before analysis, the data must be prepared using a variety of approaches. Pre-processing for image analysis may involve scaling photos to a uniform resolution, crop- ping to eliminate extraneous background, improving image quality, and lowering noise. Pre-processing for audio analysis may include normalising audio recordings, removing background noise, breaking up recordings into smaller periods, and extracting pertinent data such spectral features, pitch, and vocalisations' duration. According to the fea- tures of the data and the specifications of the analysis algorithms or models used for bird species identification, pre-processing seeks to standardise and optimise the data for accurate analysis. The precise techniques utilised may differ.

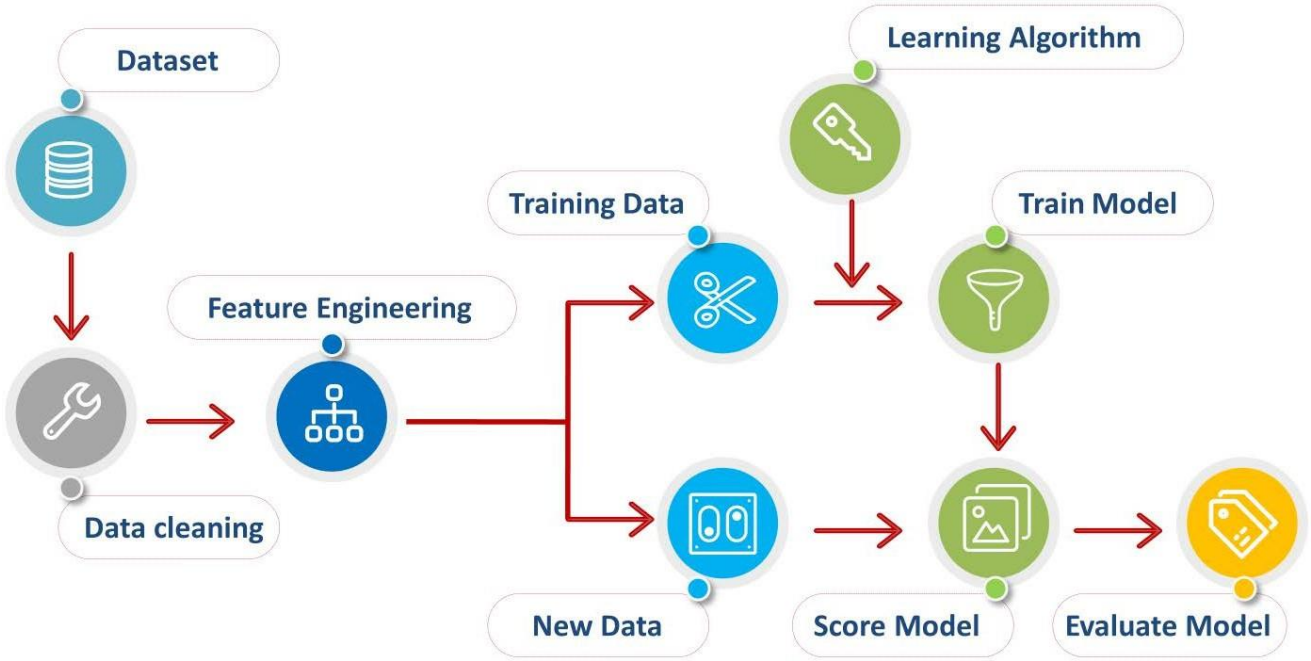

Figure 1.4: Pre-processing

1.2 Classification Methods

After the process is complete, the various classifications are taken into account to deter- mine which strategy is most popular. The operating principles of the data classifiers that were used to create and test the training and test sets will be briefly explained in these subsections.

1.2.1 CNN

In order to analyse data with a grid pattern, such as photographs, CNN is a form of deep learning model. CNN was created with the organisation of animal vision in mind and is intended to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. Convolution, pooling, and fully linked layers are the three types of layers (or ”building blocks”) that make up a standard CNN. Convolution and pooling layers in order one and two do feature extraction, whereas a fully connected layer in order three maps the extracted features into the output, such as classification. In CNN, which is made up of a stack of mathematical operations, including convolution, a specialised kind of linear operation, a convolution layer is crucial. Since a feature may appear any- where in a digital image, the pixel values are stored in a two-dimensional (2D) grid, or array of numbers, and a small grid of parameters known as the kernel, an optimizable feature extractor, is applied at each image position, making CNNs extremely effective for image processing. Extracted features may gradually and hierarchically become more sophisticated as one layer feeds its output into the following layer. Training is the process of minimising the difference between outputs and ground truth labels using an optimisa- tion technique like as backpropagation and gradient descent, among others. It involves optimising parameters such as kernels.

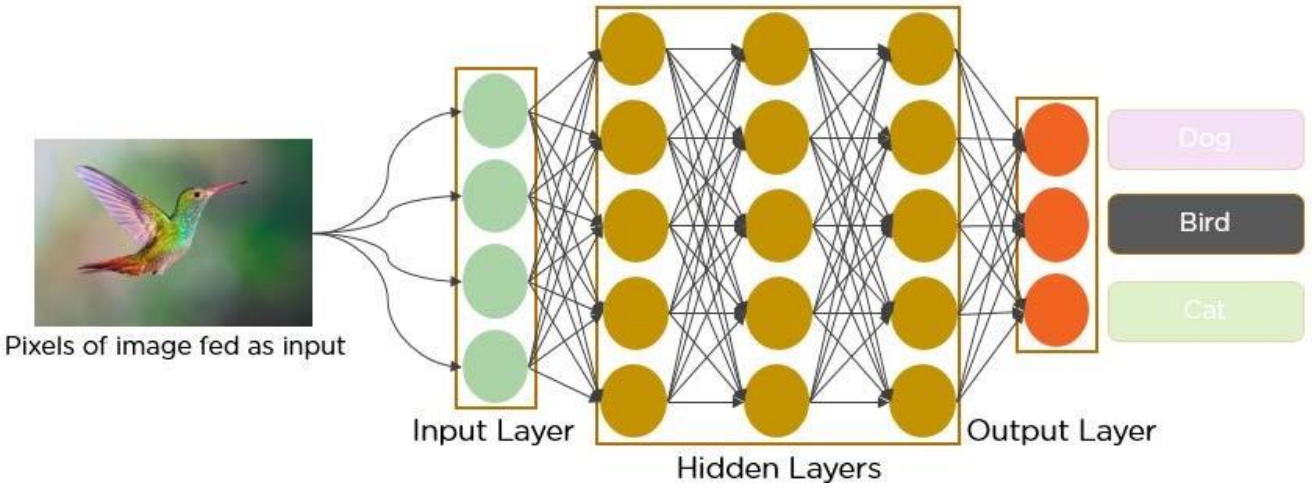

Figure 1.5: CNN

1.2.2 EfficientNet B2 Model

EfficientNet is a convolutional neural network design and scaling technique that uses a compound coefficient to consistently scale all depth, breadth, and resolution dimensions. The EfficientNet scaling method uniformly increases network breadth, depth, and resolu- tion using a set of preset scaling coefficients, in contrast to standard practise, which scales these variables arbitrarily. To employ 2N times more computing power, for instance, we may simply raise the network depth by Alpha, width by Beta, and image size by Gamma, where alpha, beta, and gamma are constant coefficients obtained by a tiny grid search on the initial small model. Network width, depth, and resolution are all consistently scaled by EfficientNet using a compound coefficient in a logical manner. The rationale behind the compound scaling method is that larger input images require more layers in order to expand the network's receptive field and more channels in order to capture more fine- grained patterns on the larger picture. The sole architectural distinction between it and the EfficientB2 model is the variation in the number of feature maps (channels), which raises the number of parameters.

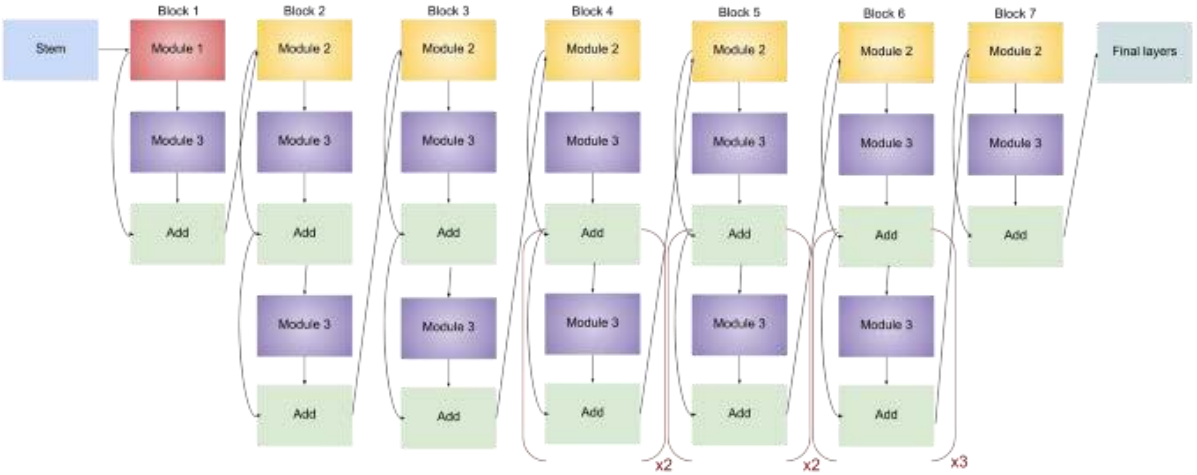

Figure 1.6: EfficientNet B2 Model

1.2.3 VGG19

In order to categorise the photos into 1000 object categories, Simonyan and Zisserman (2014) presented the VGG19, a convolutional neural network of 19 layers, 16 convolution layers, and 3 fully connected layers. The ImageNet database, which has one million photos in 1000 categories, is used to train the VGG19 algorithm. Because each convolutional layer uses numerous 3 3 filters, it is a highly well-liked technique for classifying images. According to the VGG19 architecture, the first 16 convolutional layers are utilised to extract features, and the following three layers are used to do classification. The feature extraction layers are divided into 5 groups, with a max-pooling layer coming after each group. This model receives an image with a size of 224 by 224 and produces the label of the object in the image. In the article, features are extracted using a pre-trained VGG19 model, while different machine learning approaches are used for classification. Dimensionality reduction is required to reduce the size of the feature vector because the CNN model computes a large number of parameters after feature extraction.

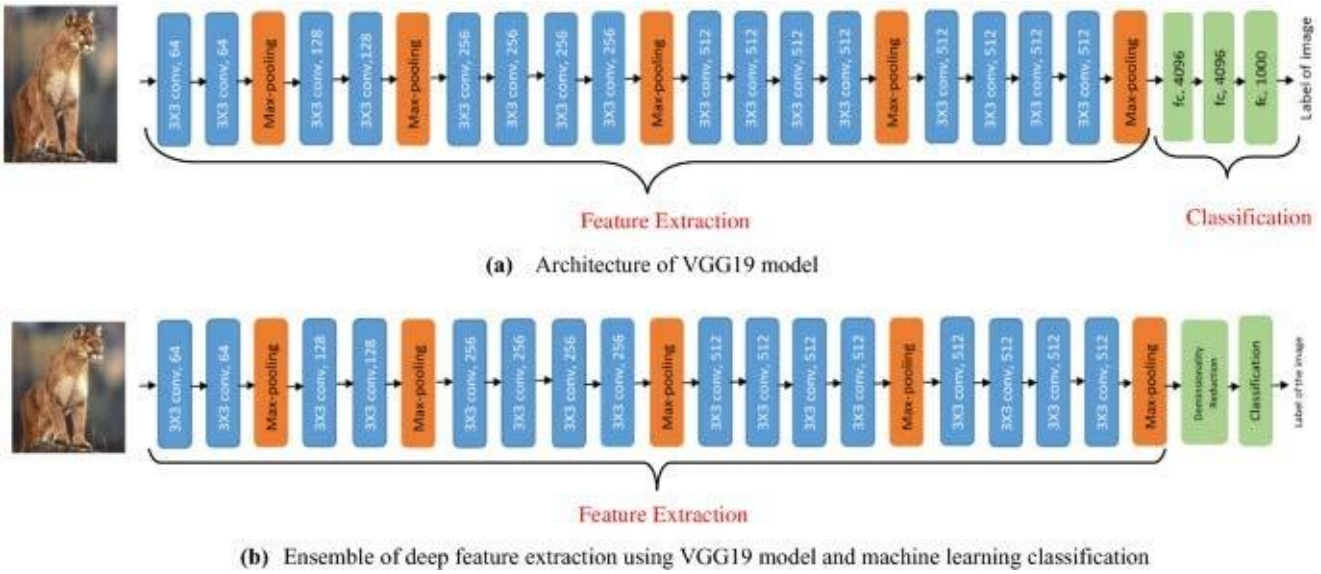

Figure 1.6: VGG19

1.3 Problem Statement

Even for seasoned birders, it can be difficult to identify a particular species. And if you've never used a field guide before, it can be difficult to know where to start looking among the hundreds of pages of species. Birds can be categorised based on certain characteristics including size, shape, and colour. We can categorise the different bird species using CNN.

Chapter 2

Datasets And Features

2.1 Datasets

2.1.1 Image Datasets

500 different bird species are included in the data set. 80,085 training images, 2500 test images (five images for each species), and 2500 validation images (five images for each species). The only bird in each image, which often occupies at least 50% of the image's pixels, is a part of this very high quality dataset. Because of this, even a somewhat sophisticated model will have training and test accuracies in the mid-90 percentile. Please note that every picture is genuine and wasn't enhanced. All pictures are in the jpg format and are 224 by 224 x 3 colours. Train, test, and validation sets are all included in the data collection. 475 subdirectories total, one for each type of bird, are included in each set. Please be aware that all of the images are real and unaltered. Each image has a resolution of 224 × 224 pixels and three colours. The data collection includes train, test, and validation sets. Each set contains a total of 475 subdirectories, one for each kind of bird. In the picture, ls. As a result, even a somewhat sophisticated model will be able to attain test and training accuracy levels in the mid-90 percent range. Please note that each photograph is unique and wasn't enhanced in any way. Every image is a 224 X 224 X 3 colour jpg file. A train set, test set, and validation set are all parts of the data set. Every set has 475 subdirectories, one for every type of bird.

NOTE: Since the test and validation photos in the data set were manually chosen as the "best" images, using those data sets as opposed to making your own test and validation sets will likely result in your model receiving the greatest accuracy score. Re- garding the model's performance on unseen images, the latter scenario is, nonetheless, more accurate.Images were found online through species-specific searches. Using a du- plicate image detection programme I created in Python, I searched the picture files for a species for duplicate photographs after downloading them. To avoid having duplicate photographs in the training, test, and validation sets, all duplicate images that were found were eliminated. The photos were then cropped so that, in most instances, the bird takes up at least 50% of each pixel. After that, the photos were converted to 224 X 224 X3 jpg files. The cropping makes sure that there is enough information in the photos for a CNN to generate a highly accurate classifier. Training, validation, and test accuracies for even a moderately robust model should be in the high 90% range. Given the size of the dataset, I advise using an image size of 150 X 150 X 3 while training a model to shorten training time. Each species' file was sequentially numbered from one throughout. These

test photos have the names 1.jpg through 5.jpg. similar for photos used for validation. Additionally, training images are consecutively numbered with ”zero” padding. Examples are 001.jpg, 002.jpg,.....010, 011,.....099.jpg, 100, 102.jpg, etc. When used with Keras flow from directory and Python file functions, the zero’s padding maintains the fileorder.

The training set is unbalanced since different species have different numbers of files. However, there are at least 130 training image files for each species. The ratio of pho- tographs of male species to images of female species is a serious flaw in the data set. About 80% of the photographs are of men, while 20% are of women. In general, males of a species have much more varied colouring than females, who are usually plain. Therefore, the appearance of male and female photographs may be completely different.The majority of test and validation photos come from the species’ male. As a result, the classifier might not perform as well on photos of female species.

2.1.2 Audio Datasets

We downloaded our dataset from xeno-canto.org using a Python wrapper from Github. It consisted of 27 selected birds (Eurasian Wren, Eurasian Jay, Common Nightingale, Tawny Owl, House Sparrow, Northern Raven, Spotted Wood-pecker, Eurasian Skylark, European Robin, Common Black-bird), each with 600 audio files, totaling 6000 data samples. Because each bird had at least 600 files that complied with our requirements, we decided to keep the number of birds to a minimum because we did not have adequate computational power to retain the data for more than 10 birds. Each sample in our data was first shorted to a duration of about 10 seconds (corresponding to a waveform length of 227.556), after which we performed feature extraction. Prior to feature extraction, we tried noise filtering using high-pass and low-pass filters, which included conducting research to determine the average frequency of our birds. However, we discovered that this range was so wide that applying these filters eliminated a significant amount of the necessary data. We chose against noise filtering as a result. Resampling our data waveforms was another goal of ours, but it required too much memory, so we abandoned that idea as well.

2.2 Feature Extraction

We mostly employed spectrograms, mel-spectrograms, and MFCC. A spectrogram is a graphic depiction of a signal’s frequencies across time. Mel-spectrograms are used to obtain a perceived scale of the audio pitches in order to have an understanding of the distances between them. They are created by converting spectrogram frequencies to the mel scale. The MFCC includes representative phonemes (different sound units) and de- scribes the general structure of the sound. Due to its ability to identify audio similarities, MFCC is frequently employed in speech recognition and information retrieval systems, such as genre

categorization. In addition to the data that these features offer, we utilised them because they transform our data into an image format that is compatible with our models. Due to a lack of resources (memory), we could only use the first channel of the audio waveforms in our models. We divided our data so that 20% was used for testing and 80% was for training. We selected this ratio because we needed a large enough sample size for our validation set to accurately reflect the performance of our models. Due to the smaller number of samples, we believed that selecting a percentage of less than 20% would cause the findings to b e skewed and deviate from the true prediction accuracy.

2.3 Performance Evaluation

Assessment metrics to gauge the effectiveness o f m achine l earning f or c lassifying bird sounds are based on confusion matrix values, a two-dimensional matrix that contains data on the actual and anticipated category.

	Actually Positive (1)	*Actually Negative (0)*
Predicted Positive (1)	True Positives (TPs)	False Positives (TPs)
Predicted Negative (0)	False Negative (FNs)	True Negative (TNs)

False alarm rate: False alarm rate is also referred to as the false positive and is defined as the proportion of Attack samples that were mistakenly forecasted to all other Normal samples.

$$\text{False Alarm Rate} = \frac{FP}{FP + TN}.$$

True negative rate: True negative rate is calculated as the proportion of correctly tagged Normal samples to all other samples.

$$\text{True Negative Rate} = \frac{TN}{TN + FP}.$$

Precision: Precision measures the proportion of attacks that are accurately anticipated to all attacks samples.

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Recall: Remember that it's the ratio of Attacks samples that are genuinely Attacks to all Attacks samples that are appropriately listed. It also goes by the name "Detection Rate."

$$\text{Recall} = \text{Detection Rate} = \frac{TP}{TP + FN}.$$

F-Measure: The harmonic mean is created by combining precision and recall. In other words, it's a mathematical technique for assessing a system's correctness while accounting for both precision and recall.

$$\text{F Measure} = 2\left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right).$$

Chapter 3

Literature Survey

3.1 Survey of Different Classification Methods

In order to detect a picture in two methods, based on feature extraction and signal clas- sification, John Martinsson et al. (2017) presented the CNN algorithm and deep residual neural networks. They conducted an experimental investigation on data sets made up of several image types. Their work, however, did not take into account the nearby species. Larger volumes of training data are needed, which might not be available, in order to identify the background species.

Convolutional neural networks trained using deep learning methods for image catego- rization were proposed by Juha Niemi, Juha T Tanttu, and colleagues (2018). It also suggested a technique for data augmentation in which photos are changed and rotated to match the desired colour. The radar-provided parameters and the image classifier's predictions are combined to form the final identification.

Based on the study of visual attributes, Li Jian, Zhang Lei, and colleagues (2014) sug- gested an efficient automatic bird species identification method. Using the similarity comparison method and the library of common photos.

Deep convolutional neural networks and data augmentation methods for audio-based bird species identification were reported by Mario Lasseck et al. in 2013. The Xeno-Canto collection of bird species audio recordings was utilised by the author in this work.

NIPS4Bplus, the first annotated, typographically enhanced bird song dataset, was pre- sented by V. Morfi et al. NIPS4Bplus consists of the bird song classification challenge dataset and tags from 2013, as well as freshly acquired temporal annotations. They have comparative information about the recordings, together with their chronological annota- tions and species-specific tags.

Pareta, A., et al.An RBF kernel function with seven input parameters and a class accu- racy feature score of 85.43 percent was used to calculate the MC-LS-VM classifier. They assert that their claims have produced the best outcomes thus far and are hence more successful thus far.

Using pre-trained CNNs and a new self-attention model that is well suited for acoustics,
K. Ko et al. came up with creative ways to finely categorise animal species based on their sound signals.
I. Lezhenin et al. proposed the LSTM model is more accurate and trustworthy than the prior model CNN and outperforms a variety of current implementations.

3.2 Existing System

There are numerous websites that use various technology to identify bird species. However, the outcomes are inaccurate. For example, let's say that when we enter information into those websites and Android applications, we get several answers rather than just the name of a single bird. It displays all of the bird names that share comparable traits. Therefore, our goal was to create a project that would result in better and more precise findings. We have classified the bird species using convolutional neural networks in order to accomplish this.

Chapter 4

Step involved in System Development Life Cycle

The System Development Life Cycle's steps are listed below. There may be numerous steps in each phase of the overall cycle.

4.1 Software Concept

Finding a need for the new system is the first step. This will involve identifying potential business opportunities or problems, completing a feasibility assessment to see if the sug- gested solution is practical, and creating a project plan. End users that have a suggestion for how to make their work better may be involved in this process. To make sure that IT is being used to support the organisation in achieving its strategic goals, the process should ideally take place concurrently with a review of the organization's strategic plan. Before any funds are budgeted for its development, management may need to accept concept ideas.

4.2 Requirement Analysis

Requirements analysis is the process of examining end users' information needs, the organ- isational

environment, and any existing systems in use in order to produce the functional specifications of a system that can satisfy users' needs. Additionally, the requirements ought to be written down in a letter, email, storyboard for the user interface, executable prototype, or any other format. To make sure the developing project is in line with user demands and requirements, the requirements documentation should be consulted throughout the remaining stages of the system development process. To guarantee that the new system will perform properly and meet end users' demands and expectations, professionals must involve end users in this process.

4.3 Architectural Design

The specifications for the hardware, software, human, and data resources, as well as the information products that will satisfy the functional requirements of the proposed system, can be established after the requirements have been established. The design will act as the system's blueprint and aid in identifying issues before they are included into the finished product. Professionals design the system, but users must examine their work to make sure it satisfies their demands.

4.4 Coding And Debugging

The process of developing the finished system involves coding and debugging. The soft- ware developer completes this phase.

4.5 System Testing

To compare the system's actual functionality to its intended or expected functionality, it must be tested. Other challenges to think about at this time include transferring old data into the new system and training staff members to use it. End users will be crucial in assessing if the produced system satisfies the required specifications and how much it is actually used.

4.6 Maintenance

The system will inevitably require maintenance. When software is provided to the cus- tomer, changes will undoubtedly be made. The change has a number of causes. Unex- pected input values entering the system could lead to change. Additionally, modifications to the system may have an immediate impact on how the software functions. The pro- gramme must to be created to support adjustments that can be made after implementa- tion. There are various software process models like:-

- Prototyping Model
- RAD Model
- The Spiral Model
- The Waterfall Model
- The Iterative Model

Of all these process models we've used the Iterative model (The Linear Sequential Model) for the development of our project.

4.7 System Development Life Cycle

The process of creating information systems through research, analysis, design, imple- mentation, and maintenance is known as the system development life cycle. Information systems development and application development are other names for the System De- velopment Life Cycle (SDLC).

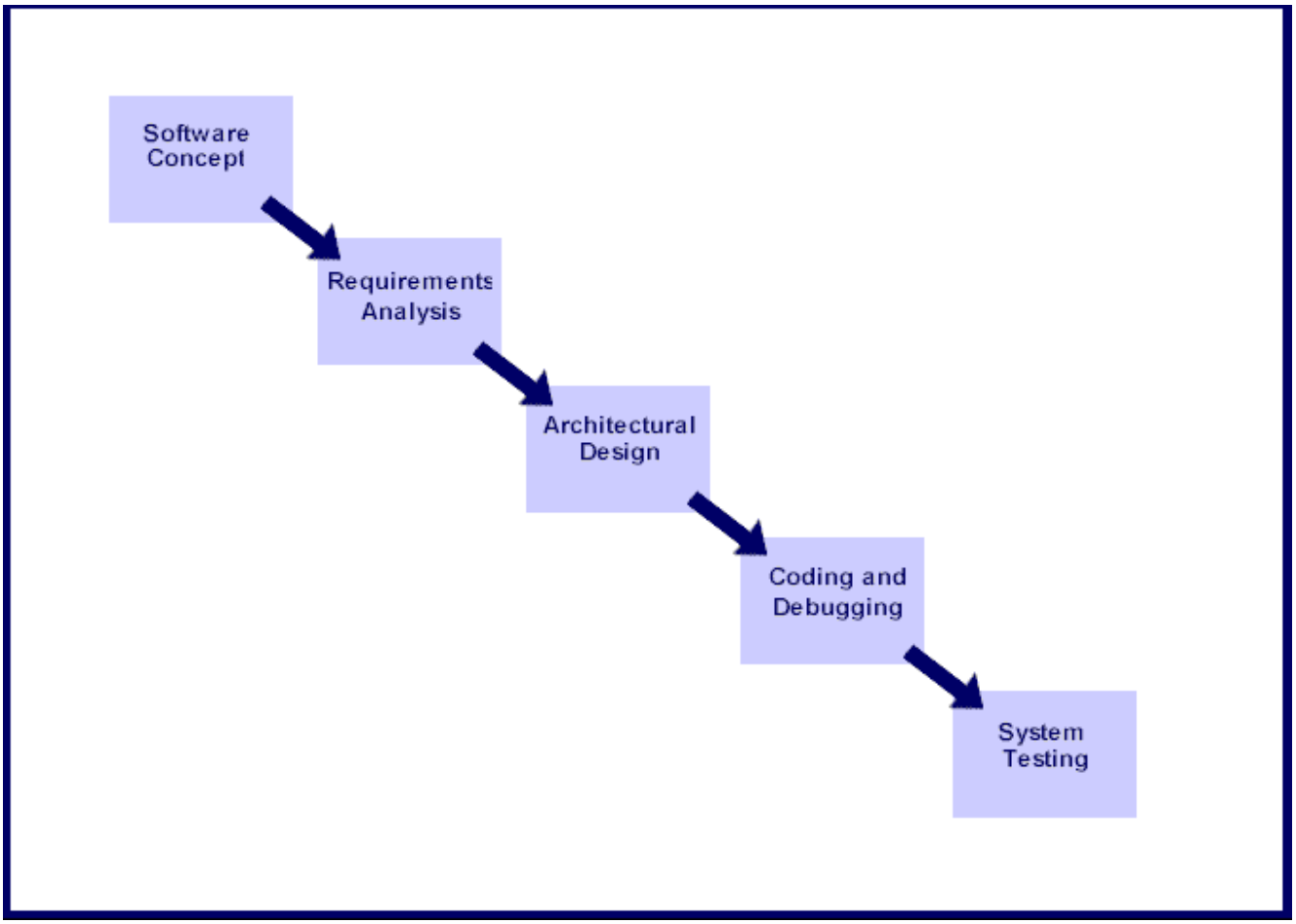

Figure 4.1: System Development Life Cycle

4.8 The Iterative Model

The cascading influence from one phase to the next, as shown in Figure, is how the waterfall model gets its name. Each phase in this paradigm has a clear beginning and end, as well as recognisable delivery to the following phase. This paradigm is also known as the software life cycle or the linear sequential model.

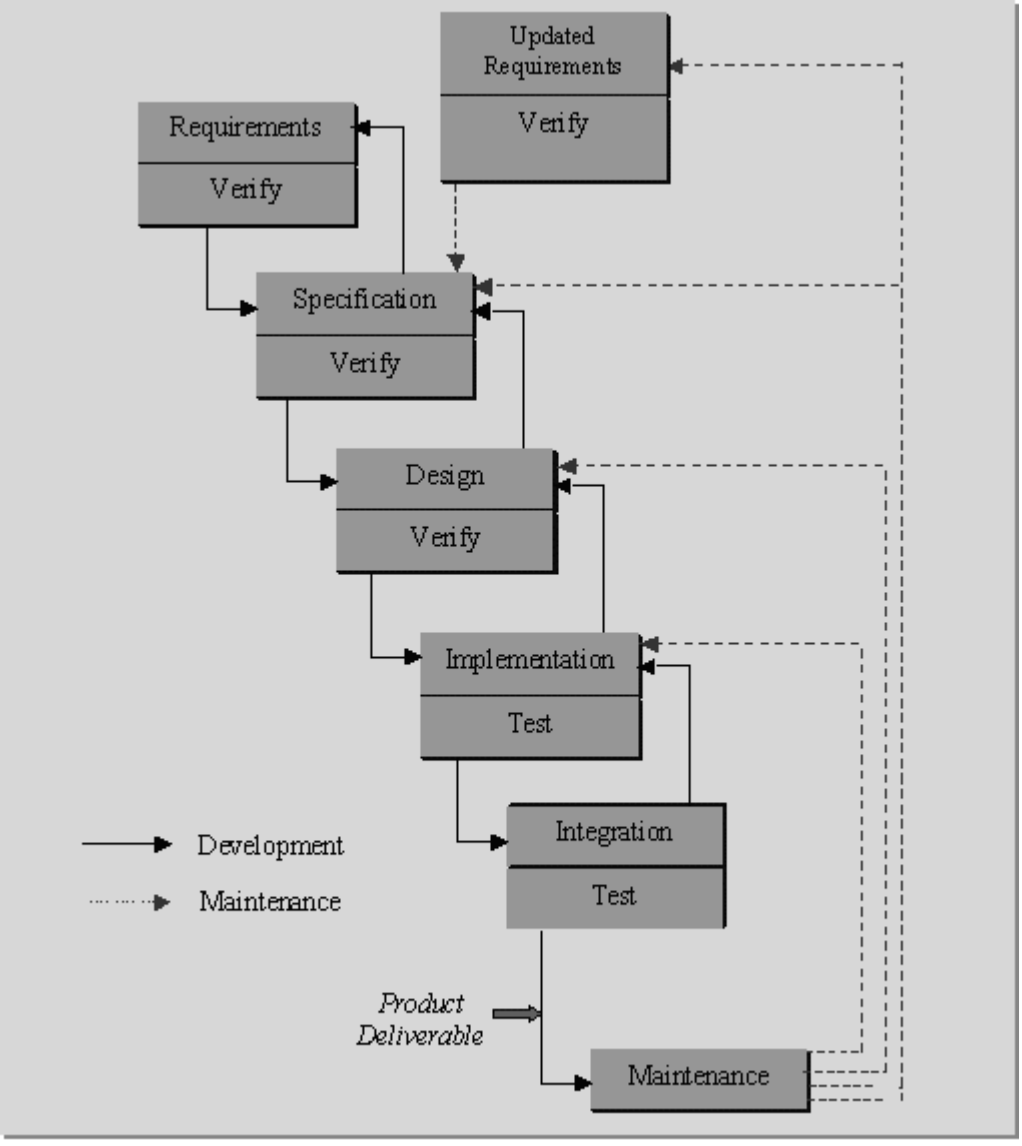

Figure 4.2: The Iterative Model

The model consists of six distinct stages, namely:

1. In the requirements analysis phase

(a) The targeted service objectives (goals) are provided along with the problem.

(b) The limitations are noted

2. The detailed definitions of (a) and (b) above are used to generate the system speci- fication during the specification process. The product function should be explicitly stated in this text.

3. The system specifications are converted into a software representation throughout the system and software design phase. The software engineer at this stage is con- cerned with:

(a) Data structure

(b) Software architecture

(c) Software architecture

(d) Algorithmic detail

(e) Interface representations

At this point, the hardware requirements are also established, along with an overview of the system architecture. The software engineer ought to be able to recognise the connections between the hardware, software, and related interfaces by the end of this phase. It is ideal to avoid passing any specification flaws ”down stream.”

4. The designs are transformed into the software domain during the implementation and testing phase.

(a) The amount of coding required can be considerably decreased by thorough documenting from the design phase.

(b) The goal of testing at this step is to find any flaws and ensure that the pro- gramme complies with the specified requirements.

5. All programme modules are integrated and tested as part of the system testing step to make sure the entire system complies with the software requirements. The software is then given to the client for acceptance testing [Deliverable - The software product is given to the client for acceptance testing].

6. The software’s maintenance phase is typically the longest. In this phase the software is updated to:

(a) Meet the evolving needs of the customer.

(b) Adapted to take into account modifications to the outside environmentadapted to take

into account modifications to the outside environment.

(c) Correct blunders and oversights that were missed during testing phases.

(d) Increasing the software's effectiveness

Be aware that feed-back loops enable the model to incorporate corrections. For instance, a design phase issue or change necessitates a'revisit' of the specifications phase. The pertinent documentation should be updated to reflect any changes made at any phase.

4.8.1 Advantages of Iterative Model

- Every stage of the iterative model includes testing by default.
- It is a strictly regimented strategy.
- It is driven by documentation, therefore documentation is created at each level.

4.8.2 Disadvantages of Iterative Model

The oldest and most popular paradigm is the waterfall model. However, many projects hardly ever adhere to its logical order. This is as a result of its strict format's inherent issues. Namely:

- Changes may be quite confusing as the project develops because iteration is only included indirectly.
- This IM has trouble addressing the initial project's inherent unpredictability because the client typically just has a general understanding of what is expected from the software solution.
- After the product has been coded, only a working version is presented to the cus- tomer. If any problems go unnoticed and progress to this point, it might be disas- trous.

Chapter 5

Proposed System

5.1 Requirement Analysis

Hardware Requirements:

- 1GB RAM
- 200GB HDD
- Intel 1.66 GHz Processor Pentium 4

Software Requirements:

- Windows XP, Windows 7,8
- Visual Studio Code
- Windows Operating System
- Python = 3.6
- Google Colab

5.2 Software And Hardware Requirements

5.2.1 Python

Python is a powerful, interactive, object-oriented, and interpreted scripting language. Python has been created to be very readable. It has fewer syntactical structures than other languages and typically employs English keywords rather than punctuation.

- The Python interpreter processes the language while it is being used. You can run your programme right away without compiling it first. This is similar to PERL and PHP.

- Python is Interactive When writing programmes, you can actually sit at a Python prompt and communicate with the interpreter directly.

- Python supports the Object-Oriented programming style or approach, which encap- sulates code within objects.

- Python is a terrific Language for Novice Programmers Python is a terrific language for

novice programmers and facilitates the creation of a variety of programmes, including simple text editors, web browsers, and games.

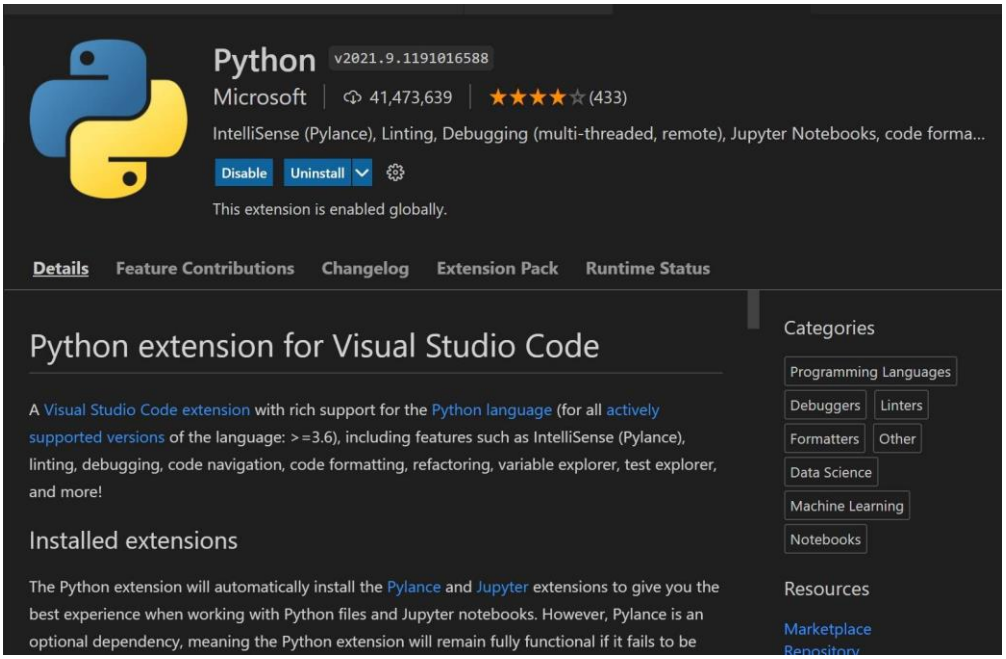

Figure 5.1: Python

5.2.2 Google Colab

In terms of AI research, Google is fairly active. Google spent many years creating the Ten- sorFlow artificial intelligence framework and Colaboratory, an application development platform. TensorFlow is now open-sourced, and Google has made Colaboratory available to everyone for free since 2017. Google Colab or just Colab are the current names for Collaboratory. The utilisation of GPU is another appealing feature that Google provides to the developers. Colab is completely free and supports GPU. Making it available to the general public for free may be done in order to establish its software as a bench- mark for university courses in data science and machine learning. It might also have a long-term goal of creating a user base for the pay-per-use Google Cloud APIs. Whatever the cause, Colab's debut has made it simpler to learn about and create machine learning applications.

Figure 5.2: Google Colab

5.2.3 Visual Studio Code

Debugging, task execution, and version control are supported by the simplified code editor Visual Studio Code. It seeks to give developers only the tools they need for a short cycle of writing code, building it, and debugging it, leaving more complex workflows to more feature-rich IDEs like Visual Studio IDE. A number of programming languages, including Java, JavaScript, Go, Node.js, Python, C++, C, Rust, and Fortran, can be used with Visual Studio Code, a source-code editor. Based on the Electron framework, which is used to create Node, it was created.

Figure 5.3: Visual Studio Code

5.3 Libraries

5.3.1 Tensorflow

Google created the open-source TensorFlow machine learning and deep learning library. It offers a versatile and effective platform for creating and refining machine learning models for a variety of applications, including speech recognition, image recognition, and natural language processing. TensorFlow is compatible with a variety of hardware setups since it supports both CPU and GPU computing.
Some key features of TensorFlow include:

- Computational Graph: TensorFlow defines and runs machine learning models using a computational graph format. This graph enables effective operation execution across various hardware, including CPUs and GPUs.

- Automatic Differentiation: TensorFlow's automatic differentiation feature makes it possible to compute the gradients required for machine learning model optimisation. Due to this, gradient-based optimisation methods, like stochastic gradient descent (SGD), are simple to construct for training models.

- Flexibility: TensorFlow provides a wide selection of APIs for creating machine learn- ing models, from high-level ones like Keras for rapid prototyping to low-level ones for detailed control over model design and training.

- Model Deployment: TensorFlow offers tools, such as TensorFlow Serving and Ten- sorFlow Lite, for deploying learned models to production contexts, enabling models to be used in production systems, mobile devices, and edge devices.

- Community and Ecosystem: TensorFlow has a sizable and active development and research community, which has produced a thriving ecosystem of deep learning and machine learning libraries, tools, and resources. Pre-trained models, visualisation tools, and tutorials for diverse use cases are all part of this.

Overall, TensorFlow is a strong and adaptable library for deep learning and machine learning that is widely utilised by academic and professional researchers to build a variety of machine learning applications.

5.3.2 Numpy

A well-liked open-source package for numerical computing in Python is called NumPy. Along with a selection of mathematical functions for quickly executing various mathemat- ical operations, it offers support for huge, multidimensional arrays and matrices. Data research, machine learning, and other scientific computing applications all make extensive use of NumPy, a core package for scientific computing in Python.

Some key features of NumPy include:

- N-dimensional arrays: Large, multi-dimensional data arrays can be effectively han- dled by NumPy thanks to the useful array object ndarray. NumPy's ndarray ob- ject offers a number of array operations, including indexing, slicing, reshaping, and broadcasting, which makes it simple to use arrays for sophisticated computations.

- Mathematical functions: A wide range of mathematical operations, including el- ementary arithmetic, linear algebra, Fourier analysis, random number generation, and more, are available in NumPy. These performance-optimized functions offer a quick and effective way to run mathematical operations on data arrays.

- Interoperability: With other Python numerical computing libraries like SciPy, scikit- learn, and TensorFlow, NumPy supports interoperability. It is a versatile tool for data analysis and scientific computing workflows because it also enables integration with other data processing libraries.

- Performance:Since NumPy is written in C and offers optimised array operations, numerical computations can be completed quickly and effectively. Because of its effectiveness, NumPy is frequently utilised in scientific computing applications that call for the effective management of big datasets and challenging mathematical computations.

- Community and Ecosystem: The enormous and vibrant community of NumPy users and developers has produced a thriving ecosystem of libraries, tools, and resources for Python's numerical computation. Libraries for visualisation, linear algebra, sta- tistical analysis, and other topics are included in this, greatly enhancing NumPy's capability and potential.

Providing effective array operations and mathematical functions for a variety of scien- tific computing tasks, NumPy is a strong and crucial toolkit for numerical computing in Python. Due to its performance, flexibility, and vast ecosystem, it is frequently used in applications for data research, machine learning, and scientific computing.

5.3.3 Keras

Python is supported by the open-source neural network library known as Keras. It is well- liked for its modularity, quickness, and simplicity of usage. As a result, it can be utilised for both quick experimentation and quick prototyping. Convolutional neural networks, recurrent neural networks, and both are supported for implementation. On the CPU and GPU, it is capable of operating without any hiccups. In contrast to libraries that are more extensively used, such as TensorFlow and Pytorch, Keras offers user-friendliness that enables users to quickly create neural networks without spending too much timeon technical jargon.

5.3.4 Pandas

An open-source library for high-performance data processing in Python is known as Python Pandas.In Python, t is employed for data analysis. Processing steps including merging, cleansing, and restructuring are all necessary for data analysis. For quick data processing, a variety of tools are available, including Numpy, Scipy, Cython, and Panda. However, we like Pandas since they are quicker, easier, and more expressive to use than other tools. Given that Pandas is built on top of the Numpy package, Numpy is neces- sary in order to use Pandas. Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps re- quired for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

5.4 Flow of System

The actual flow of the suggested system is depicted in the image below. A training dataset is needed to classify a picture in order to create such a system. Both train and test outcomes are included in the trained dataset. The image is briefly kept in the database each time a user uploads an input file. The system then feeds this input file to CNN, which is connected to a trained dataset. Convolutional layers of several types make up CNN. To achieve the highest level of accuracy, a variety of elements, including the head, colour, body, shape, and the overall image of the bird, are taken into account. Each alignment is passed through a deep convolutional network in order to extract features from the network's various levels. The classification of the image is then performed using CNN and an unsupervised process known as deep learning. Furthermore, the image is categorised pixel by pixel using a grey scale approach. After that, these features are combined and sent to the classifier. In order to produce potential outcomes, the input will be compared to the trained dataset in this case. An autograph is created during categorization, and it contains of nodes that eventually connect to form a network. A score sheet is prepared based on this network, and output is produced with the aid of the score sheet.

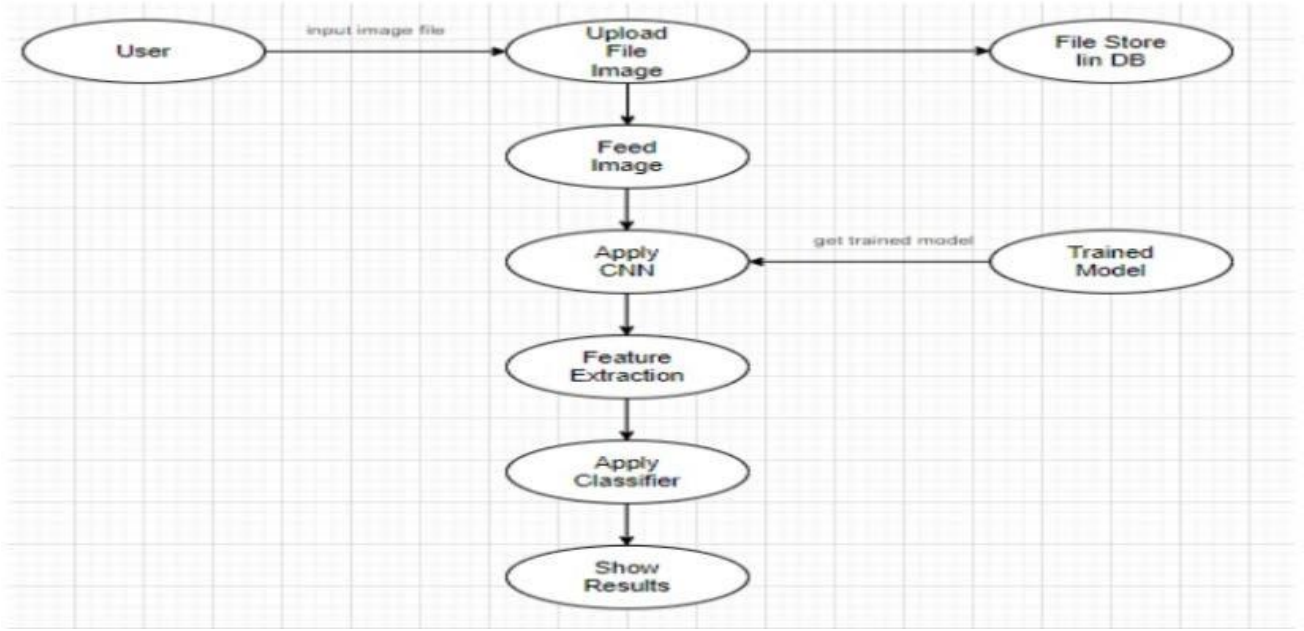

Figure 5.4: Flow of System

5.5 Results

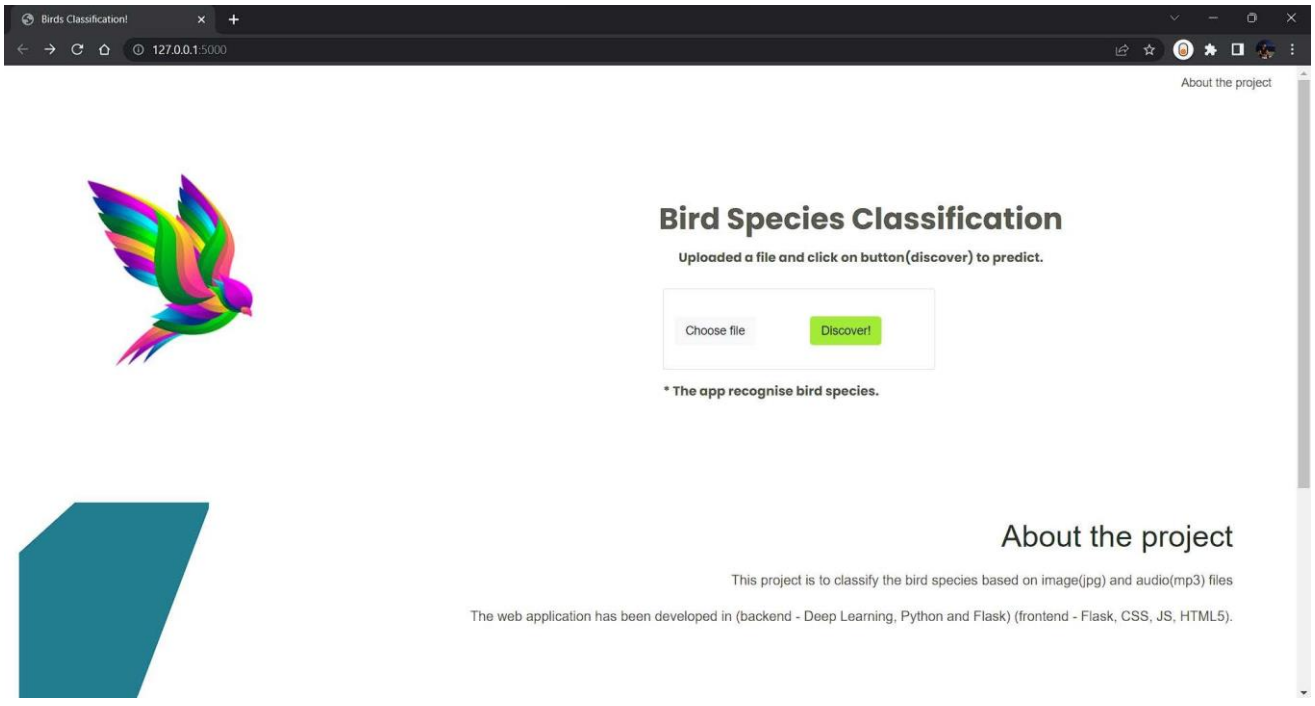

Figure 5.5: The UI/ Homepage of the Project

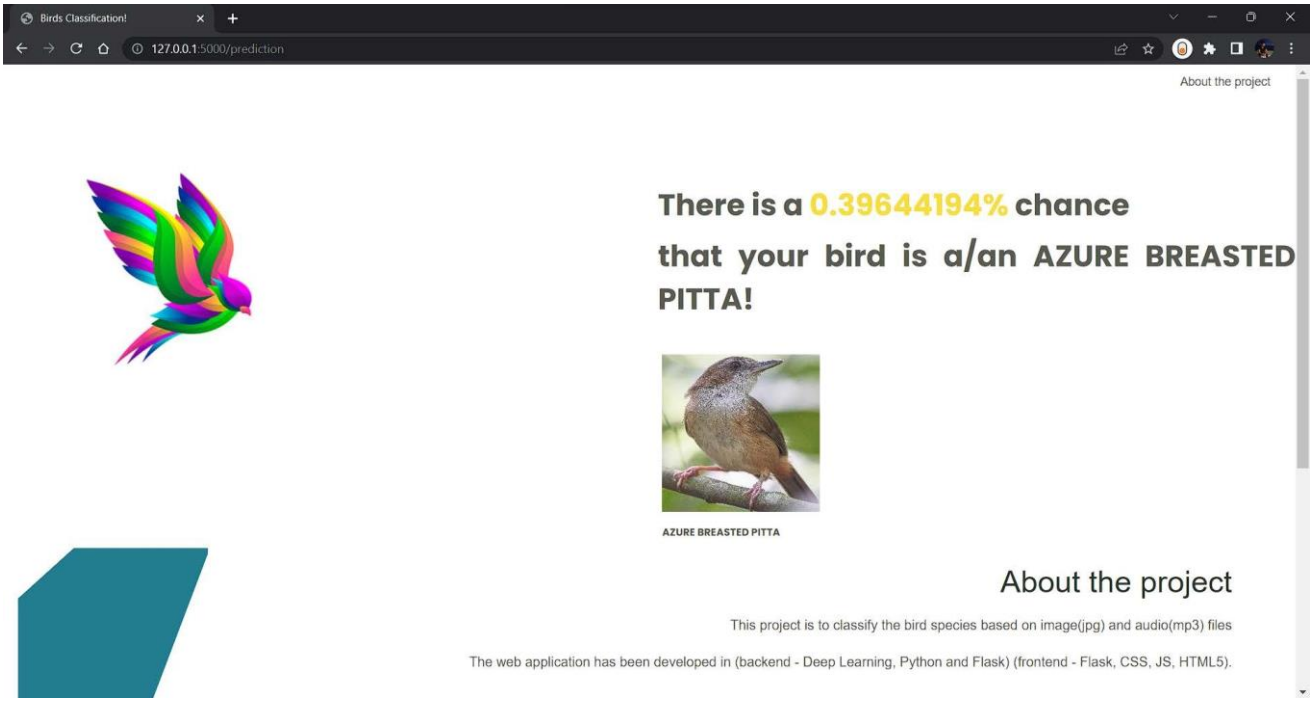

Figure 5.6: Identifying Bird through Image

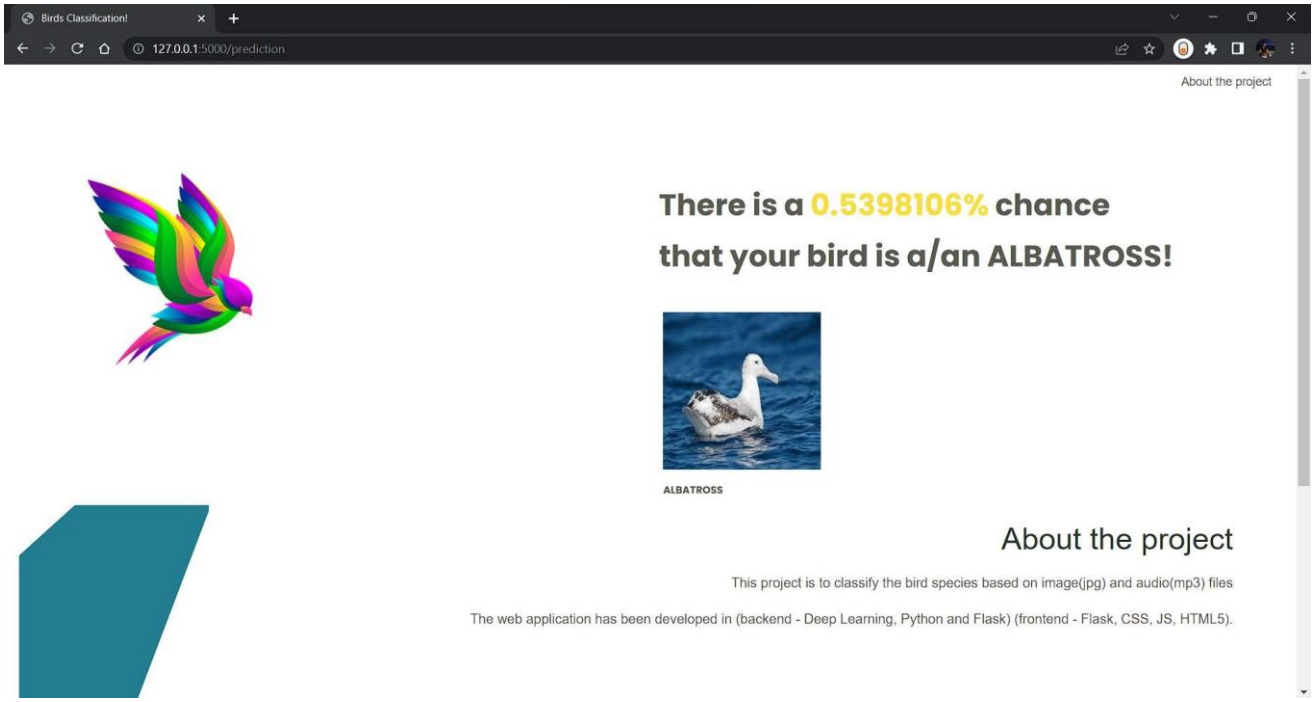

Figure 5.7: Identifying Bird through Image

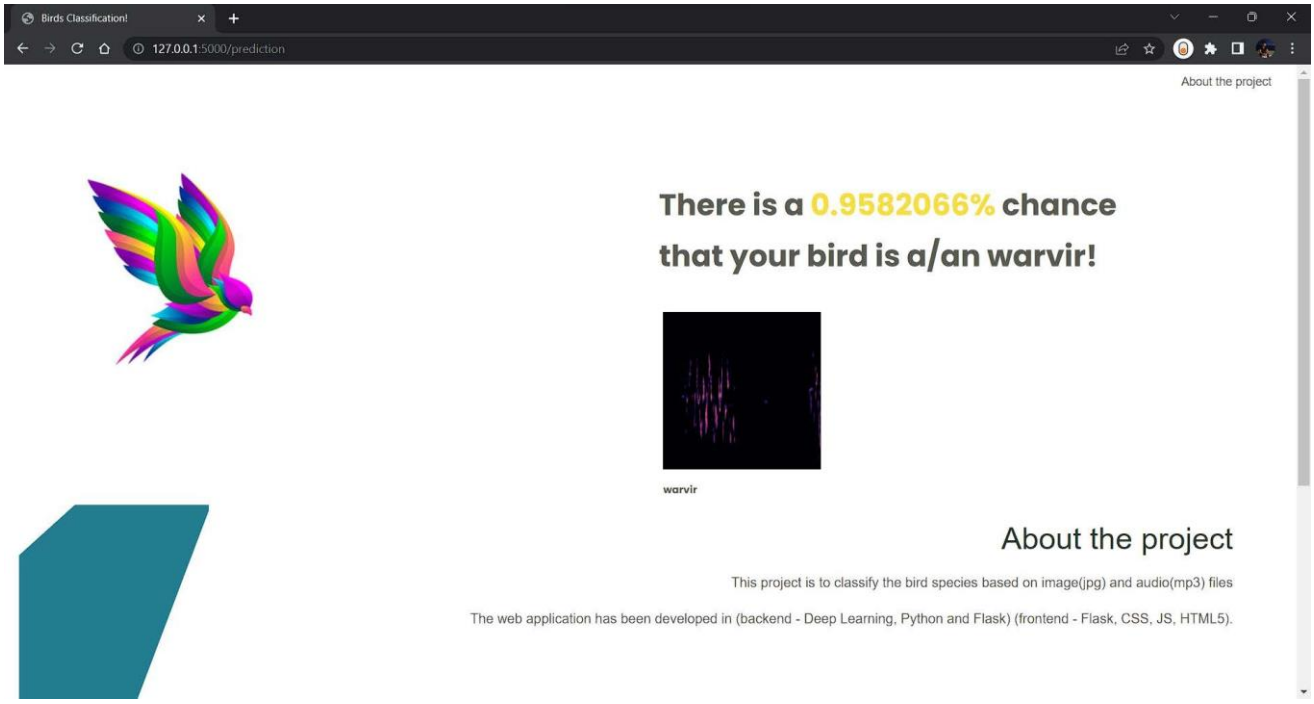

Figure 5.8: Identifying Bird using Audio

Chapter 6

Conclusion

Overall, we attempted to accurately classify birds using audio data from xeno-canto by im- plementing Classification model and adapting CNN models from past work. We achieved varying success with multiple models, including CNN Classification and Transfer learning Models. Similarly, for image data from Kaggle, we implemented transfer learning Classi- fication model and adapted CNN models to achieve the best accuracy. Throughout the process, we discussed our prior research and methodology for preprocessing our specific dataset.

Bibliography

[1] Charbuty B, Abdulazeez A. "Classification Based on Decision Tree Algorithm for Machine Learning," J. Appl. Sci. Technol. Trends. 2021;01:20–28, 2021.

[2] Saleem SI, Abdulazeez AM, Orman Z. "A New Segmentation Framework for Arabic Handwritten Text Using Machine Learning Techniques; 2021.

[3] Sukri MMM, Fadlilah U, Saon S, Mahamad AK, Som MM, Sidek A. "Bird Sound Identification based on Artificial Neural Network," 2020 IEEE Student Conf. Res. Dev. SCOReD. 2020.

[4] Madhavi A, Pamnani R, Student PG. "Deep Learning Based Audio Classifier for Bird Species. 2018;S3(10):228–233.

[5] Gerry, "325 bird species-classification." 2022, Accessed: Feb. 10, 2021. [Online]. Avail- able: https://www.kaggle.com/gpiosenka/100-bird-species.

[6] Kahl S, Wilhelm-Stein T, Klinck H, Kowerko D, Eibl M. Recognizing birds from sound - The 2018 BirdCLEF baseline system," arXiv; 2018.

[7] Xu W, Zhang X, Yao L, Xue W, Wei B. "A multi-view CNN-based acoustic clas- sification system for automatic animal species identification," Ad Hoc Networks. 2020;102:102115.

[8] Koh CY, Chang JY, Tai CL, Huang DY, Hsieh HH, Liu YW. "Bird sound classification using convolutional neural networks," CEUR Workshop Proc. 2019;2380.

[9] Acevedo MA, Corrada-Bravo CJ, Corrada- Bravo H, Villanueva-Rivera LJ, Aide TM. "Automated classification of bird and amphibian calls using machine learning: A comparison of methods," Ecol. Inform. 2009;4(4):206–214.

[10] Qian K, Zhang Z, Baird A, Schuller B. "Active learning for bird sound classification via a kernel-based extreme learning machine," J. Acoust. Soc. Am. 2017.