DEEP LEARNING CONVOLUTION NEURAL NETWORK FOR LARVAE IMAGE CLASSIFICATION

Ms.B.Akhila[1], Gogula Abhinaya[2], Allampati Geetha[2], Bellapa Jyothi[2], Chinta Pavani[2], Bommisetty Bharathi[2]

[1]Department of Electronics and Communication Engineering, Geethanjali Institute of Science and Technology, Nellore.

ABSTRACT: The classification of larvae images is a vital task across multiple domains, including biology, ecology, and agriculture. Traditional methods employed for larvae image classification often rely on manual feature extraction and handcrafted algorithms, resulting in time-consuming processes and limitations in effectively handling large datasets. Such methods may struggle with variations in lighting, scale, and orientation, leading to reduced classification accuracy. Additionally, adapting traditional techniques to different larvae species can be challenging and require expert knowledge. This research presents the development and implementation of a Convolutional Neural Network (CNN) based on deep learning for larvae image classification. The proposed deep learning CNN architecture aims to surmount the limitations of existing systems by autonomously learning discriminative features from larvae images. This approach reduces the reliance on manual feature engineering and enhances the model's ability to accommodate diverse datasets and various species. Through extensive training on labeled data, the CNN model becomes proficient in accurate and efficient larvae image classification. The proposed system offers a scalable and adaptable solution for researchers and practitioners in the field, contributing to advancements in larvae-related studies and applications.

Keywords: Larvae Image Classification, Convolutional Neural Network (CNN), Deep Learning, Traditional Methods, Manual Feature Extraction, Handcrafted Algorithms, Variations In Lighting, Scale, And Orientation, Discriminative Features, Labeled Data, Scalable And Adaptable Solution, Biology, Ecology, And Agriculture, Diverse Datasets.

1. INTRODUCTION

The study of larvae, a critical developmental stage in many animals, particularly insects, offers fascinating insights into the life cycle, adaptation, and survival strategies of these organisms. During the larval stage, organisms undergo significant transformations, preparing them for adulthood. This period is characterized by rapid growth and, in many cases, a feeding frenzy to accumulate the necessary energy for metamorphosis. The diversity of larval forms and behaviors reflects the wide range of ecological niches they occupy, from leafy branches inhabited by caterpillars to the underwater realms of mosquito and dragonfly larvae.

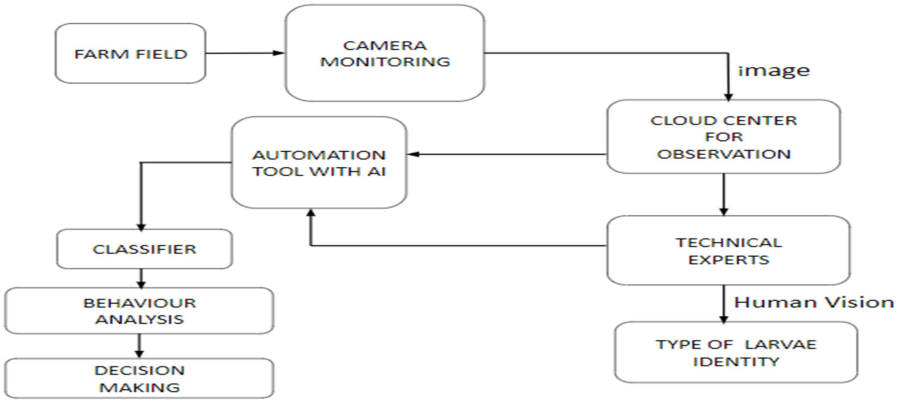

Figure1.1 Research Motivation

The motivation behind designing a deep learning Convolutional Neural Network (CNN) based classification system, particularly for the context of agriculture and pest management, is deeply rooted in the necessity to enhance productivity and sustainability in farm fields. With the advent of precision agriculture, camera monitoring systems have become an indispensable asset, serving as the eyes on the field that continuously capture detailed images of crops and their potential pests, including various types of larvae. These images are then transmitted to a cloud center for observation, where they can be processed and analyzed at scale. Herein lies the critical role of technical experts who leverage deep learning models, such as CNNs, to develop sophisticated classifiers capable of identifying specific types of larvae from the captured images. The identification process is not merely about naming the larvae; it extends to understanding their behavior through behavior analysis. This insight is crucial for decision-making processes concerning pest management strategies.The primary research objective for utilizing Deep Learning Convolutional Neural Networks (CNNs) for Larvae Image Classification is to innovate and refine computational models that can automatically and accurately classify various larvae species from images. This research aims to push the boundaries of current machine-learning techniques by developing CNN architectures specifically optimized for the unique challenges presented by larvae imagery, such as variability in size, shape, color, and texture among species. A significant part of this endeavor involves the creation and annotation of a comprehensive and diverse dataset that encompasses a wide range of larvae species under various environmental conditions. The goal is to enhance the CNN's ability to learn and generalize from this dataset, ensuring high accuracy across unseen images and real-world scenarios.

2. LITERATURE SURVEY

Saeed et. al [1] proposed a model that could be used on a system for the detection of mosquito larvae in order to eliminate its habitat. The model was developed from scratch and obtained a training accuracy of 93.95% and testing accuracy of 90.18%. Upon further testing with 100 images, the accuracy was found to be 86.0% and precision was 92.2%.

Hong et. al [2] proposed a deep learning model that was able to identify the presence of mosquitoes when attached to autonomous robots traveling in drains and alert authorities of the locations of identified breeding sites. This model could also be implemented into other tools such as extendable sticks for NEA officers to use during home inspections.

Martins et. al [3] developed an image classification model that could differentiate between Ae. Aegypti, Aedes albopictus, and Culex sp. larvae using pictures taken with a cellphone camera by comparing various Deep Learning models (Mobilenetv2, ResNet18, ResNet34, EfficientNet_B0,and EfficientNet_Lite0).The best results were obtained with EfficientNet_Lite0 with an accuracy of 97.5% during validation and 90% during testing, an acceptable result considering the risks related to misclassification in this context. These results demonstrated the viability of classifying mosquito larvae, differentiating even between Aedes species, and thus contributed to the prevention of dengue.

Lee et. al [4] developed an automatic image analysis method to identify mosquito species using a deep learning-based object detection technique. Color and fluorescence images of live mosquitoes were acquired using a mosquito capture device and were used to develop a deep learning-based object detection model. Among the deep learning-based object identification models, the combination of a swine transformer and a faster region-convolutional neural network model demonstrated the best performance, with a 91.7% F1-score. This indicates that the proposed automatic identification method could be rapidly applied for efficient analysis of species and populations of vector-borne mosquitoes with reduced labor in the field .

Defèr et. al [5] developed an intelligent bee colony system equipped with sensors, actuators, and robots was used to optimally manage and guide the bee colony through contemporary challenges. Part of the research within the Hiveopolis project dealt with automated methods for monitoring the brood nest on a honeycomb, which is useful for assessing the colony strength. This thesis leveraged high-resolution image data of a honey bee colony, recorded with the hive observation setup, from the BeesBook project at the Biorobotic Lab, whose team also contributed to Hiveopolis, at the Freie Universität Berlin, in order to investigate how well it was possible to predict honey bee brood age with high-resolution image data. The developed deep learning system served as a baseline for a reference system, which would provide ground truth data for teaching more inexpensive, non-invasive, and space-saving machine learning systems, such as temperature sensor-based systems, with the sensors installed on the back of the honeycomb. Since, with a prediction error of 2.1±26.5 hours (mean±standard deviation) and a mean absolute error of 0.66 days, the system did not operate at reference level. It was found that even if the image data was highly resolving, the system would benefit from cameras whose focus was set on the interior of the cells and from improved lighting of the inside of the cells. But also from human-performed quality control of the automatically generated training data set.

Gunes et. al [6] developed a study to detect larvae, ideal for the production of royal jelly. Initially, a camera setup capable of taking clear photographs of honeycomb cells was prepared. With this setup, photographs of honeycombs containing larvae of different sizes were taken. Later, the larvae with the ideal size in the photographs were labeled, and the convolutional neural network was trained. Finally, honeycomb cells and center points were identified with the Hough circle, and the locations of the larvae according to the honeycomb cell were determined. In conclusion, a system that could successfully identify ideal-sized larvae and their locations to be used in the production process for royal jelly was created.

Chowdhury et. al [7] proposed a research to classify the larvae of invasive species like Zebra and Qua gga mussels, which are native to Eastern Europe but invasive in United States waterways. It is crucial to identify invasive species at the larval stage when they are mobile in the water and before they have established a presence, to prevent infestations. Video-based underwater species classification faces several challenges due to

illumination variations, angle of view, and background noise. For invasive larvae, the added difficulty comes from their microscopic size and the minor differences between aquatic species larvae. Furthermore, data imbalance presents a challenge, as invasive species are typically less abundant than native species. In video-based surveillance methods, each organism may be represented in multiple video frames, offering different views that show various angles, conditions, etc. Given the multiple images per organism, the research proposed using image set-based classification, which can accurately classify invasive and non-invasive organisms based on sets of images. This approach often yields higher accuracy, even if the accuracy of single image classification is lower. The system classifies image sets using a feature-averaging pipeline that begins with an autoencoder to extract features from the images. These features are then averaged for each set corresponding to a single organism. A classifier trained on the image set features makes the final prediction. The experiments demonstrated that feature averaging significantly improves over other models of image classification, achieving more than 97% F1 score in predicting invasive organisms in video imaging data for a Quagga mussel survey.

Đurđević et. al [8] developed a model in which they examined the use of geometric morphometric analysis (GMA), deep learning (Convolutional Neural Networks), and computer vision (deep CNN) applied to the mouthparts (mandibles) of chironomid larvae as a proxy for identifying the relationship between the functional morphology and food acquisition behavior. They determined the variability in morphology of mandibles for 23 taxa of chironomid larvae from different genera, subfamilies, and their Functional Feeding Group (FFG). Analysis using GMA showed that the five different FFGs examined had mandibular traits that significantly varied in shape and size. A deep CNN model was then built that could classify the 23 taxa into their respective FFG automatically with 92.31% accuracy. A gradient-weighted Class Activation Mapping (Grad-CAM) algorithm found that the most important parts of mandibles for classification were the gula and mandibular joint. They introduced three additional species to the deep CNN models to test whether automatic classification would directly and automatically identify traits of the specimens independently from taxonomic identification. The deep CNN process avoids issues surrounding both taxonomic identification and previous knowledge of a specific taxon's feeding trait, and in all cases, the model classified taxa correctly based on their mandibular traits. The use of deep learning approaches could substantially enhance the use of trait-based approaches and increase the reliability and use of chironomids in bioassessment.

Xu et. al [9] designed a portable image acquisition device using a mobile phone with a macro lens to collect 1st-6th instar larval images. The YOLOv4 detection method and improved MRES-UNet++ segmentation methods were used to locate the larvae and segment the background. The larval length and head capsule width were automatically measured by some graphics algorithms, and the larval image features were extracted by SIFT descriptors. The random forest model, improved by Boruta feature selection and grid search method, was used to identify the larval instars of FAWs. The test results showed that high-definition images could be easily collected by using the portable device (Shenzhen, China). The MRES-UNet++ segmentation method could accurately segment the larvae from the background. The average measurement error of the head capsule width and body length of moth larvae was less than 5%, and the overall identification accuracy of 1st–6th instar larvae reached 92.22%. His method provides a convenient, intelligent, and accurate tool for technicians to identify the larval instars of FAWs.

WRB Bessa et. al [10] developed a solution based on the performance evaluation of a set of counting models for use in embedded structures. Additionally, this application can be scalable for counting different species and sizes. Besides, the dataset named Vivarium and its specifications is presented as proof of concept and used in the model evaluation. Results showed that the prediction model based on convolutional neural networks was capable of verifying the compliance of images, achieving an accuracy of 99%.

3. PROPOSED SYSTEM

3.1 OVERVIEW

In recent years, the application of deep learning techniques, particularly Convolutional Neural Networks (CNNs), has revolutionized various domains, including image classification. One of the fascinating areas where CNNs have demonstrated remarkable performance is in the classification of larvae images. Larvae, being a crucial stage in the life cycle of many organisms, pose unique challenges for classification due to their diverse morphological features and subtle differences between species. Accurate and efficient classification of larvae images holds immense significance in various fields, including biology, ecology, agriculture, and environmental science.

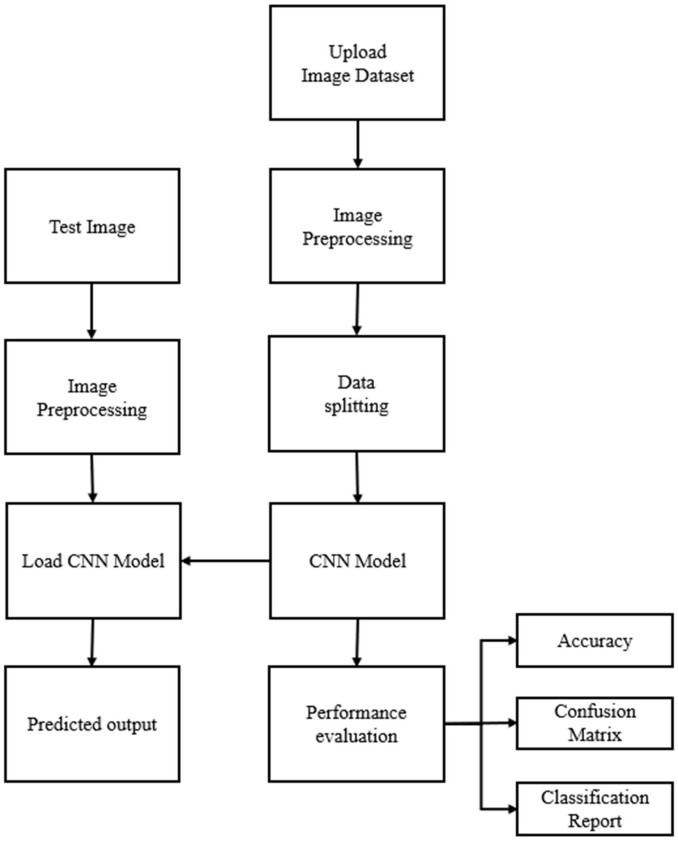

Figure : Proposed Block diagram of Larvae Image classification

This paper aims to provide a comprehensive introduction to the utilization of CNNs for larvae image classification. We begin by discussing the importance of larvae classification and the challenges associated with

traditional methods. Subsequently, we delve into the fundamentals of CNNs, elucidating their architecture, components, and functioning principles. Understanding CNNs is pivotal for comprehending their effectiveness in processing larvae images and extracting discriminative features.

Furthermore, we explore the pre-processing techniques tailored for larvae image data, including data augmentation, normalization, and dimensionality reduction. These preprocessing steps are crucial for enhancing the robustness and generalization ability of CNN models, especially when dealing with limited training data or imbalanced classes.

The core of this paper lies in the discussion of various CNN architectures optimized for larvae image classification. We present a comparative analysis of state-of-the-art CNN architectures, such as VGG 16, and Inception, highlighting their strengths and weaknesses concerning larvae image classification tasks. Additionally, we discuss transfer learning strategies, where pre-trained CNN models on large-scale datasets like ImageNet are fine-tuned for larvae classification tasks, reducing the need for extensive training data and computational resources.

Moreover, we shed light on recent advancements and innovations in CNN architectures specifically tailored for larvae image classification. These include attention mechanisms, capsule networks, and graph-based CNNs, which aim to capture intricate spatial dependencies and hierarchical structures present in larvae images more effectively. Finally, we discuss evaluation metrics and methodologies for assessing the performance of CNN models in larvae image classification tasks. We emphasize the importance of robust evaluation practices to ensure reliable and reproducible results.

3.2 Image Preprocessing

In deep learning CNN models for larvae image classification, image preprocessing plays a crucial role in enhancing model performance and robustness. Here are some common preprocessing steps:

Image Resizing:

Images in the dataset may come in various sizes. Resizing them to a consistent size is essential for ensuring uniformity and compatibility with the CNN model's input dimensions. Typically, images are resized to a square shape, often based on the input size expected by the CNN architecture (e.g., 224x224 pixels for models like AlexNet or VGG).

Normalization:

Normalizing pixel values helps in stabilizing training and improving convergence. Common normalization techniques involve scaling pixel values to a specific range, such as [0, 1] or [-1, 1]. This step is crucial for ensuring that the input features have similar scales, which aids in faster convergence during training.

Data Augmentation:

Data augmentation techniques are employed to increase the diversity and size of the training dataset, thereby reducing overfitting and improving generalization. Augmentation techniques may include random rotations, flips, shifts, zooms, and changes in brightness or contrast. For larvae image classification, augmentation can simulate variations in lighting conditions, orientations, and backgrounds.

Data Balancing:

In datasets where certain classes are underrepresented (e.g., certain species of larvae occur less frequently), data balancing techniques can be applied to ensure that the model does not bias towards the majority class. This can

be achieved through techniques such as oversampling, undersampling, or more advanced methods like synthetic data generation.

3.3 Data Splitting

Data splitting is a crucial step in training deep learning CNN models for larvae image classification. It involves partitioning the available dataset into separate subsets for training, validation, and testing. Here are the typical steps involved in data splitting:

Initial Data Preparation:

Ensure that your dataset containing larvae images is properly labeled with the corresponding classes. Each image should be associated with a label indicating the species or category of larvae it belongs to.

Define the Split Ratios:

Determine the proportions in which you want to split your dataset into training, validation, and testing sets. Common split ratios include 70% for training, 15% for validation, and 15% for testing. However, these ratios can vary depending on factors such as the size of the dataset and the complexity of the task.

Shuffle the Dataset:

Before splitting the data, it's essential to shuffle the dataset to ensure that the data samples are randomly ordered. This helps prevent any bias that may arise if the data samples are ordered based on certain criteria (e.g., class labels).

Split the Dataset:

Divide the shuffled dataset into three subsets: training, validation, and testing. This can be done using various methods, such as manual splitting, built-in functions provided by deep learning frameworks (e.g., TensorFlow's train_test_split), or custom scripts.

Training Set:

The training set is used to train the CNN model. It should contain the majority of the dataset, typically around 70-80%. The model learns from the images and their corresponding labels in this set.

Validation Set:

The validation set is used to tune hyperparameters, monitor the model's performance during training, and prevent overfitting. It helps in selecting the best-performing model by providing an unbiased evaluation. The validation set usually comprises around 10-20% of the dataset.

Testing Set:

The testing set is used to evaluate the final performance of the trained model on unseen data. It serves as an independent benchmark to assess the model's generalization ability. The testing set should not be used during model training or hyperparameter tuning. It typically constitutes the remaining portion of the dataset (around 10-20%).

Check Class Distribution:

After splitting the dataset, it's important to verify that each subset (training, validation, testing) maintains a similar distribution of class labels as the original dataset. This ensures that the model learns from a representative sample of each class and that the evaluation results are reliable.

Save the Split Datasets:

Save the split datasets (images and corresponding labels) into separate directories or files. This facilitates easy access during model training, validation, and testing phases.

Data Augmentation (Optional):

If data augmentation techniques are employed during training (e.g., random rotations, flips, shifts), they should be applied only to the training set to prevent data leakage and ensure unbiased evaluation on the validation and testing sets.

By following these steps, you can properly split your dataset into training, validation, and testing subsets, ensuring robust training, reliable model evaluation, and accurate assessment of the CNN model's performance for larvae image classification tasks.

3.4 CNN Model

Convolutional Neural Networks (CNNs) have emerged as a powerful tool for image classification tasks due to their ability to automatically learn features from raw data. Unlike traditional methods that require handcrafted features, CNNs can automatically learn hierarchical representations directly from the data. In this paper, we delve into the intricacies of CNN classification models, exploring their architecture, training methodology, and practical applications.

3.4.1 CNN Layers

In a deep learning CNN (Convolutional Neural Network) model designed for larvae image classification, the architecture typically consists of several layers, each serving a specific purpose in extracting features from the input images and making predictions. Here's an overview of the key components and concepts involved:

Input Layer:

The input layer of the CNN receives the raw pixel values of the input images. The size of the input layer is determined by the dimensions of the input images (e.g., width, height, and number of color channels).

Convolutional Layers:

Convolutional layers are the core building blocks of CNNs. They consist of multiple filters (also known as kernels) that slide over the input image, performing element-wise multiplications and summations to produce feature maps. These filters capture local patterns and spatial relationships in the images, enabling the network to learn hierarchical representations of features.

For inputs to the CNN, the depth is the number of channels in the image (i.e., a depth of three when working with RGB images, one for each channel). For volumes deeper in the network, the depth will be the number of filters applied in the previous layer.

To make this concept more clear, let’s consider the forward-pass of a CNN, where we convolve each of the K filters across the width and height of the input volume. More simply, we can think of each of our K kernels sliding across the input region, computing an element-wise multiplication, summing, and then storing the output value in a 2-dimensional activation map.

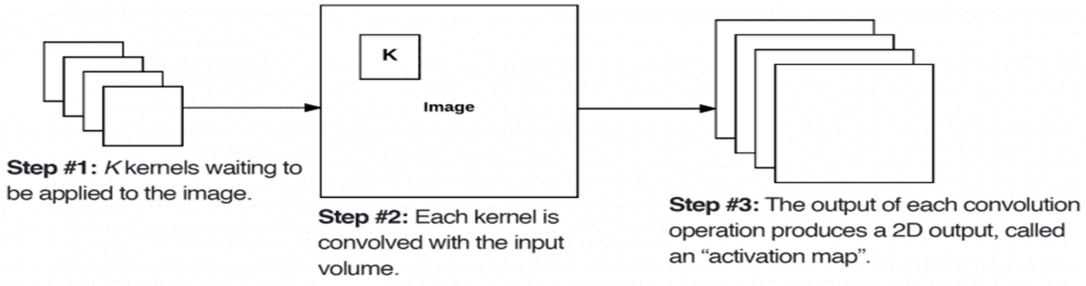

figure 4.1 The pipeline of the general CNN architecture

After applying all K filters to the input volume, we now have K, 2-dimensional activation maps. We then stack our K activation maps along the depth dimension of our array to form the final output volume

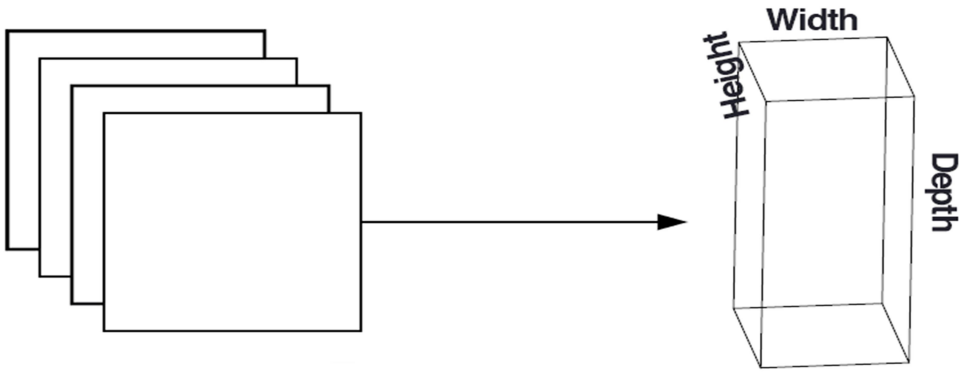

figure 4.2 stacking of Activation map

Figure 2: After obtaining the K activation maps, they are stacked together to form the input volume to the next layer in the network.

Activation Function:

Typically, each convolutional layer is followed by an activation function such as ReLU (Rectified Linear Unit). The activation function introduces non-linearity into the network, enabling it to learn complex relationships between features.

ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.

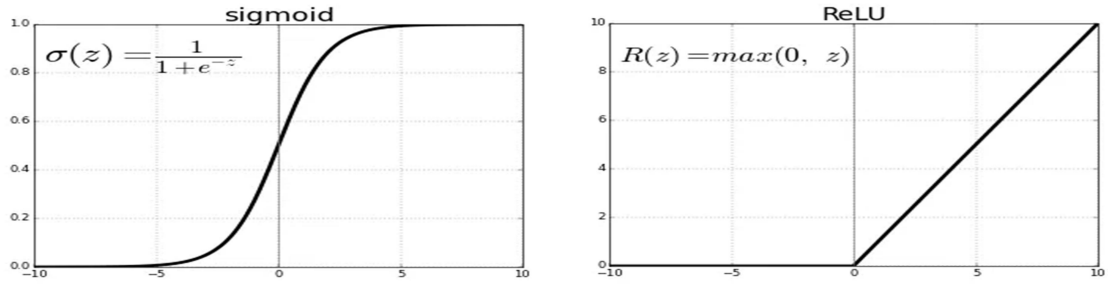

figure 4.3 Activation function used between the hidden layers.a)Sigmoid,b)ReLu

As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: [0 to infinity)

The function and its derivative both are monotonic.

But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

Pooling Layers:

Pooling layers are used to reduce the spatial dimensions of the feature maps while retaining the most important information. Max pooling, for example, selects the maximum value from a region of the feature map, effectively downsampling it.

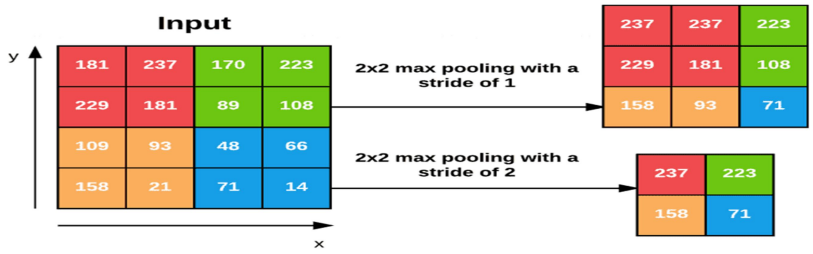

Figure 4.4 Max pooling Layer

We can further decrease the size of our output volume by increasing the stride — here we apply S = 2 to the same input (Figure , bottom). For every 2×2 block in the input, we keep only the largest value, then take a step of two pixels, and apply the operation again. This pooling allow us to reduce the width and height by a factor of two, effectively discarding 75% of activations from the previous layer.

In summary, POOL layers Accept an input volume of size Winput×Hinput×Dinput. They then require two parameters:

The receptive field size F (also called the “pool size”). The stride S.

Applying the POOL operation yields an output volume of size Woutput×Houtput×Doutput, where:

Woutput = ((Winput −F) / S) +1

Houtput = ((Hinput −F) / S) +1

Doutput = Dinput

Fully Connected Layers:

After several convolutional and pooling layers, the feature maps are flattened into a vector and passed through one or more fully connected (dense) layers. These layers perform high-level reasoning and decision-making based on the extracted features.

Output Layer:

The output layer of the CNN produces the final predictions. For larvae image classification, the output layer typically consists of one neuron per class, with a softmax activation function to output probabilities indicating the likelihood of each class.

3.4.2 Loss Function:

During training, the CNN computes a loss function to measure the difference between the predicted probabilities and the actual labels. Common loss functions for classification tasks include categorical cross-entropy and binary cross-entropy.

A loss function is a function that compares the target and predicted output values; measures how well the neural network models the training data. When training, we aim to minimize this loss between the predicted and target outputs.

The hyperparameters are adjusted to minimize the average loss — we find the weights, wT, and biases, b, that minimize the value of J (average loss).

$$J(w^T, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

Optimization Algorithm:

The optimization algorithm (e.g., Stochastic Gradient Descent, Adam) is used to minimize the loss function by adjusting the weights of the network through backpropagation. This process involves computing the gradients of the loss with respect to the network parameters and updating the parameters in the direction that minimizes the loss.

Hyperparameters:

Hyperparameters such as learning rate, batch size, number of layers, filter sizes, and dropout rate are crucial settings that need to be tuned to optimize the performance of the CNN model.

Regularization Techniques:

Techniques like dropout and weight regularization (e.g., L1 or L2 regularization) may be employed to prevent overfitting and improve the generalization ability of the model.Another strategy to regularize deep neural networks is dropout. Dropout falls into noise injection techniques and can be seen as noise injection into the hidden units of the network.In practice, during training, some number of layer outputs are randomly ignored (dropped out) with probability p.

During test time, all units are present, but they have been scaled down by p. This is happening because after dropout, the next layers will receive lower values. In the test phase though, we are keeping all units so the values will be a lot higher than expected. That's why we need to scale them down.By using dropout, the same layer will alter its connectivity and will search for alternative paths to convey the information in the next layer. As a result, each update to a layer during training is performed with a different "view" of the configured layer. Conceptually, it approximates training a large number of neural networks with different architectures in parallel.

"Dropping" values means temporarily removing them from the network for the current forward pass, along with all its incoming and outgoing connections. Dropout has the effect of making the training process noisy. The choice of the probability p depends on the architecture.

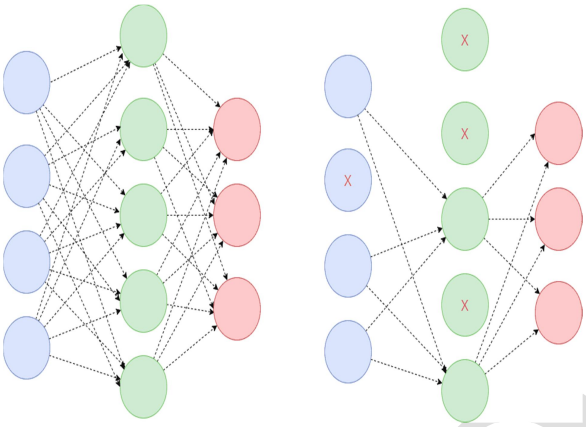

figure 4.5 Regularization techniques for training deep neural networks

3.4.3 Training Process:

Training a CNN involves optimizing its parameters (weights and biases) to minimize a predefined loss function. The process typically involves the following steps.

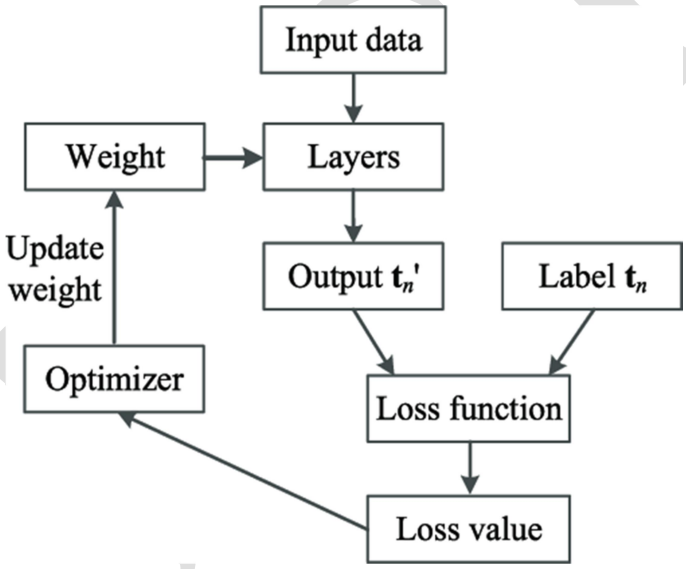

figure 4.6 Schematic Diagram of Training Process of the CNN

Forward Propagation: During forward propagation, input data is passed through the network, and predictions are made based on the current set of parameters. Now, let us see the neural network structure to predict the class for this binary classification problem. Here, I am going to use one hidden layer with two neurons, an output layer with a single neuron and sigmoid activation function.

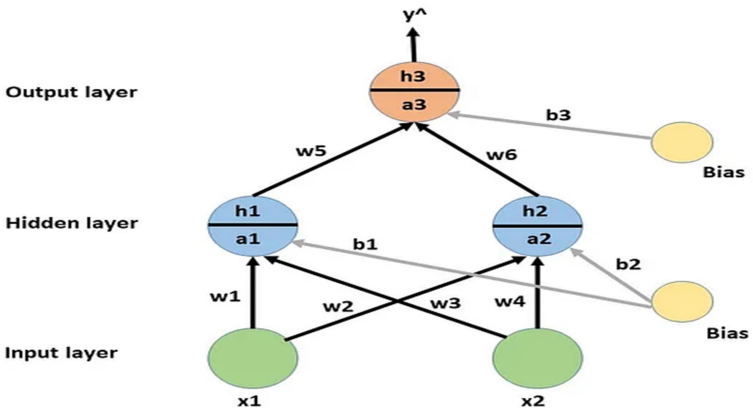

figure 4.7 Forward propagation in neural networks

During forward propagation at each node of hidden and output layer preactivation and activation takes place. For example, at the first node of the hidden layer, a1(preactivation) is calculated first and then h1(activation) is calculated.

a1 is a weighted sum of inputs. Here, the weights are randomly generated.

a1 = w1*x1 + w2*x2 + b1 = 1.76* 0.88 + 0.40*(-0.49) + 0 = 1.37 approx and h1 is the value of activation function applied on a1.

$$h1 = \frac{1}{1+e^{-a1}} = 0.8 \text{ approx.}$$

Similarly

a2 = w3*x1 + w4*x2 + b2 = 0.97 *0.88 + 2.24 *(- 0.49)+ 0 = -2.29 approx and

$$h2 = \frac{1}{1+e^{-a2}} = 0.44 \text{ approx.}$$

For any layer after the first hidden layer, the input is output from the previous layer.

a3 = w5*h1 + w6*h2 + b3 = 1.86*0.8 + (-0.97)*0.44 + 0 = 1.1 approx

and

$$h3 = \frac{1}{1+e^{-a3}} = 0.74 \text{ approx.}$$

So there are 74% chances the first observation will belong to class 1. Like this for all the other observations predicted output can be calculated.

Loss Computation: A loss function is used to measure the disparity between the predicted outputs and the ground truth labels.

$$MSE = \frac{1}{N}\sum_{i}^{N}(Yi - \hat{Y}i)^2$$

Backward Propagation (Backpropagation): Backpropagation calculates the gradient of the loss function with respect to each parameter in the network. This gradient is then used to update the parameters in the direction that minimizes the loss.For backpropagation there are two updates performed, for the weights and the deltas. Lets begin with the weight update.

We are looking to compute which can be interpreted as the measurement of how the change in a single pixel in the weight kernel affects the loss function .

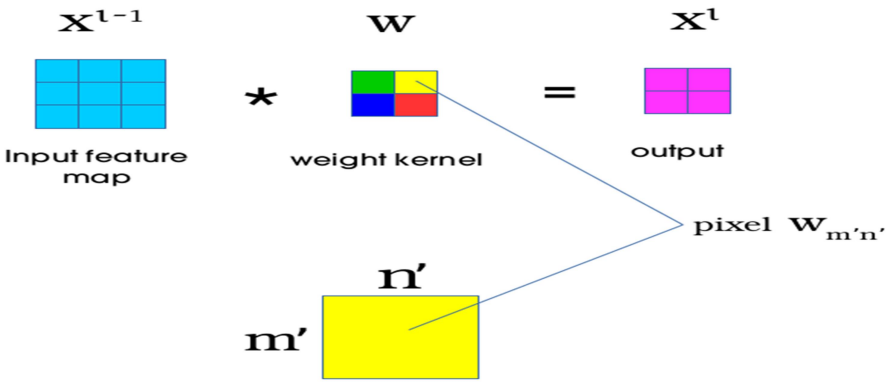

figure 4.8 Backpropagation in neural networks

Parameter Update: The parameters (weights and biases) of the network are updated using optimization algorithms such as Stochastic Gradient Descent (SGD), Adam, or RMSprop.

Iterative Optimization: The process of forward propagation, loss computation, backward propagation, and parameter update is repeated iteratively until convergence or a predefined stopping criterion is met.

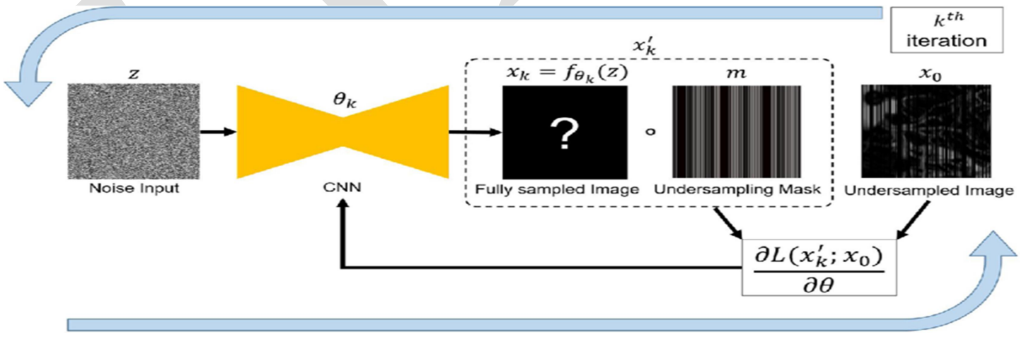

figure 4.9 Schematics of the iterative DIP optimization

3.5 Advantages

No Human Supervision: This means that the system operates autonomously without requiring continuous human intervention. It can analyze, process, and make decisions without the need for human oversight. This could involve tasks such as image recognition, where the system identifies objects or patterns in images without human guidance.

High Accuracy at Image Recognition: This refers to the system's ability to accurately identify objects, patterns, or features within images. High accuracy means that the system can correctly recognize and classify objects with a high degree of precision, minimizing errors or misclassifications.

Minimize Computation: This involves optimizing the computational resources required by the system to perform tasks such as image recognition. Minimizing computation ensures efficient use of computing resources, reducing processing time and energy consumption.

Same Knowledge Across All Image Locations: This means that the system's knowledge or understanding of the images is consistent and uniform across different locations or contexts. Regardless of where the image is taken, the system applies the same set of knowledge or rules for analysis and recognition**.**

4. RESULTS AND DISCUSSION

4.1 IMPLEMENTATION DESCRIPTION

1. GUI Setup with Tkinter: The code initializes a Tkinter window (`main`) with a title ("LARVAE CLASSIFICATION") and a specified geometry. It also defines various fonts and buttons for user interaction, such as uploading a dataset, image processing, training models, making predictions, viewing performance graphs, and exiting the application.

2. Global Variables: Several global variables are declared to hold important data like the dataset's file path (`filename`), input and output data for models (`X`, `Y`), the trained model (`model`), and performance metrics (`accuracy`, `p1`, `r1`, etc.).

3. Larvae Class Labels: A predefined list (`shapes`) contains the names of larvae classes. A function (`getID(name)`) maps the class names to numeric labels for model training.

4. Dataset Upload (`uploadDataset`): This function allows users to select a dataset directory through a file dialog. The path of the selected directory is stored, and a message is displayed in the GUI's text area.

5. Image Processing (`imageProcessing`): Though the core functionality (image loading, resizing, and normalization) is commented out, the expected behavior is to read images from the uploaded dataset, preprocess them, and split them into training and test sets.

6. Gaussian Naive Bayes Classifier (`gnb`): This function either loads preprocessed data from `.npy` files or processes raw images from a dataset directory. It trains a Gaussian Naive Bayes classifier on the training data and evaluates it on the test data, displaying metrics like accuracy, precision, recall, F1 score, and a confusion matrix.

7. CNN Model Training (`cnnModel`): This function either loads a pre-trained CNN model or defines and trains a new one. The CNN architecture includes convolutional and max pooling layers, followed by dense layers for classification. The model is trained on the training data and evaluated on the test data, with performance metrics displayed similarly to the GNB classifier.

8. Prediction (`predict`): Allows users to upload an image and classify it using the trained CNN model. The predicted class is displayed on the image.

9. Performance Comparison Graph (`accuracy_comparison_graph`): This function generates a bar graph comparing the performance metrics (precision, recall, F1 score, and accuracy) of the GNB classifier and the CNN model.

10. Closing the Application (`close`): Closes the GUI window.

The GUI provides a user-friendly interface for performing complex tasks like training machine learning models and classifying images with minimal technical knowledge required from the user. It demonstrates the integration of machine learning with GUI programming to create practical and interactive applications.

4.2 RESULTS AND DESCRIPTION

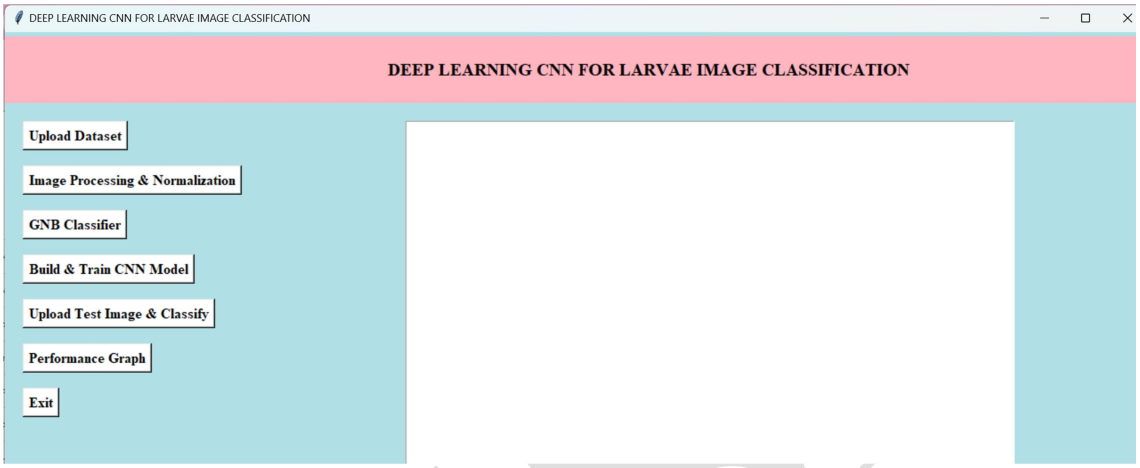

figure 4.1 Deep Learning CNN for Larvae image Classification

The above Python application implements a Tkinter-based GUI application for classifying larvae images using deep learning CNN. It allows users to upload datasets, pre-process data, train various classification models (Guassian Naive bayes, CNN)

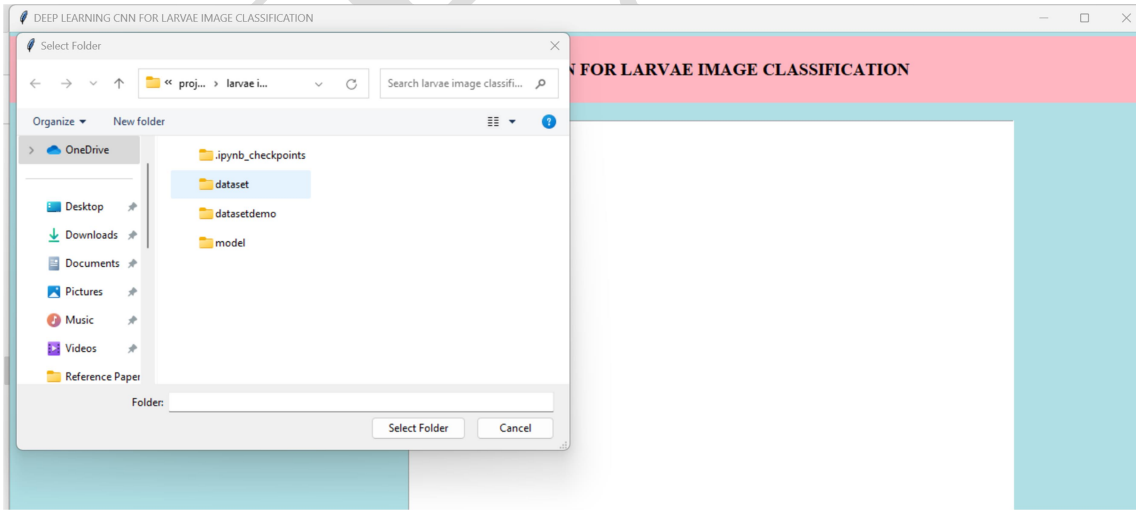

Figure 4.2 Uploading the dataset for larvae image classification

The application enables users to upload larvae image datasets for analysis. Upon selection, the dataset undergoes pre-processing. Additionally, users can upload test data to assess model performance and classify different types of larvae based on the trained models.

Figure 4.3 After uploading the dataset

Once the larvae dataset is uploaded, the application pre-processes the dataset. This ensures data readiness for model training.

Figure 4.4 After pre-processing the uploaded dataset

The pre-processed dataset exhibits enhanced cleanliness and organization, ready for analysis or model training. It has undergone necessary transformations to ensure optimal quality and usability.

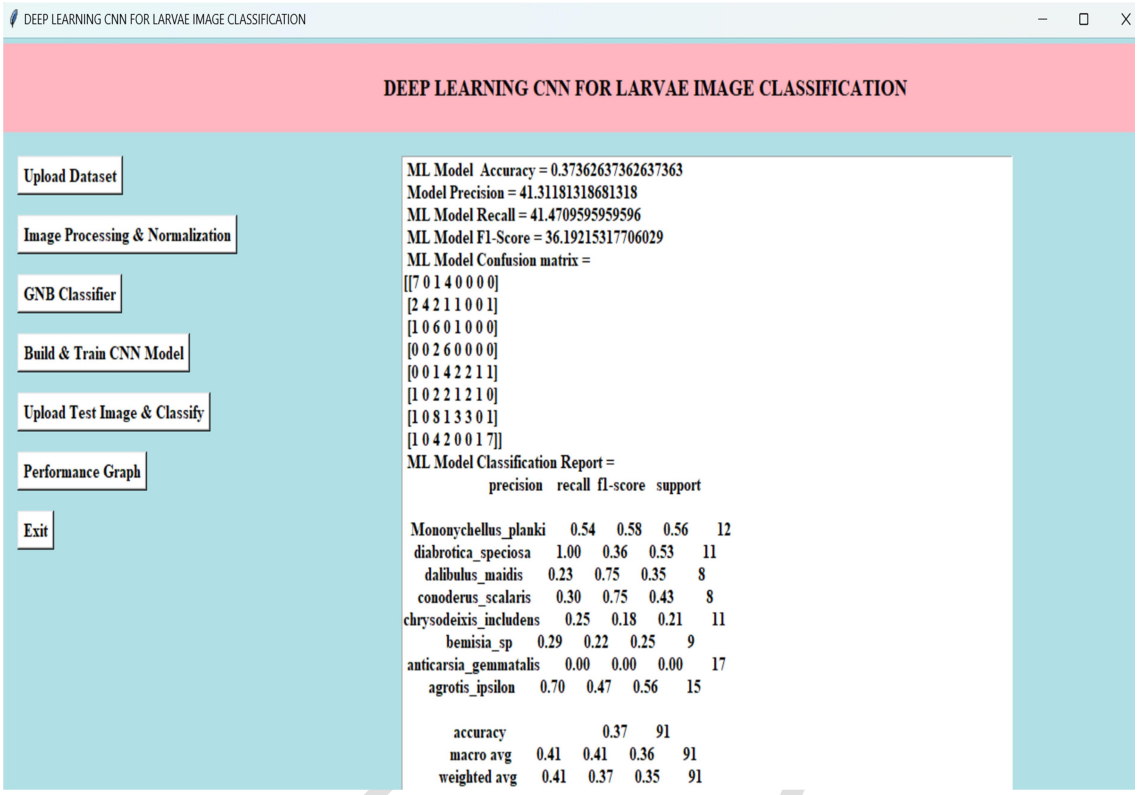

Figure 4.5 Performance of GNB Classifier

Table 4.2 Classification Report of GNB Classifier

S.No	Classes Names	Precision	Recall	F1-score	Support
1	Mononychellus_planki	0.54	0.58	0.56	12
2	diabrotica_speciosa	1.00	0.36	0.53	11
3	dalibulus_maidis	0.23	0.75	0.35	8
4	conoderus_scalaris	0.30	0.75	0.43	8
5	chrysodeixis_includeus	0.25	0.18	0.21	11
6	bemisia_sp	0,29	0.22	0.25	9
7	anticarsia_gemmatalis	0.00	0.00	0.00	17
8	agrotis_ipsilon	0.70	0.47	0.56	15
	accuracy			0.37	91
	macro avg	0.41	0.41	0.36	91
	weighted avg	0.41	0.37	0.35	91

4.3.4: Built & Train CNN Model

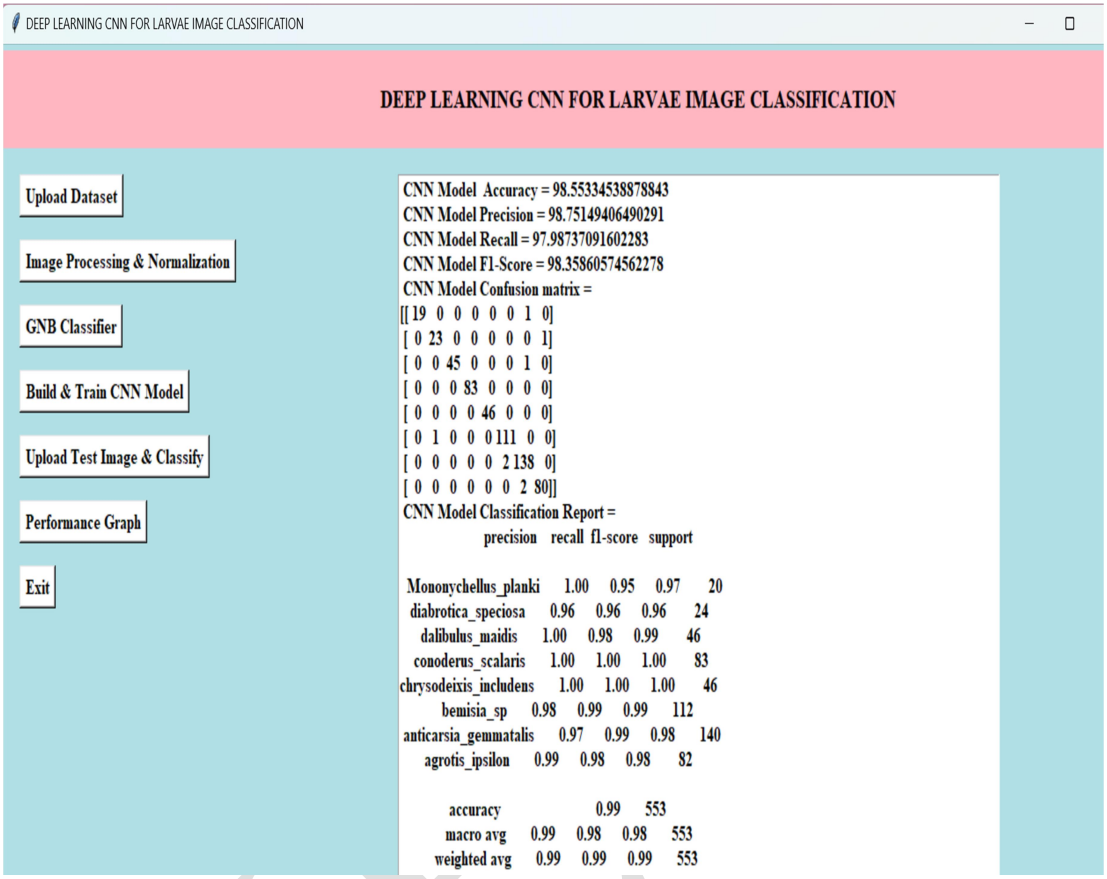

Figure 4.6 Performance of CNN Model

Table 4.3 Classification Report of CNN Model

S.No	Classes Names	precision	recall	f1-score	support
1	Mononychellus_planki	1.00	0.95	0.97	20
2	diabrotica_speciosa	0.96	0.96	0.96	24
3	dalibulus_maidis	1.00	0.98	0.99	46
4	conoderus_scalaris	1.00	1.00	1.00	83
5	chrysodeixis_includeus	1.00	1.00	1.00	46
6	bemisia_sp	0.98	0.99	0.99	112
7	anticarsia_gemmatalis	0.97	0.99	0.98	140
8	agrotis_ipsilon	0.99	0.98	0.98	82
	accuracy			0.99	553
	macro avg	0.99	0.98	0.98	553
	weighted avg	0.99	0.99	0.99	553

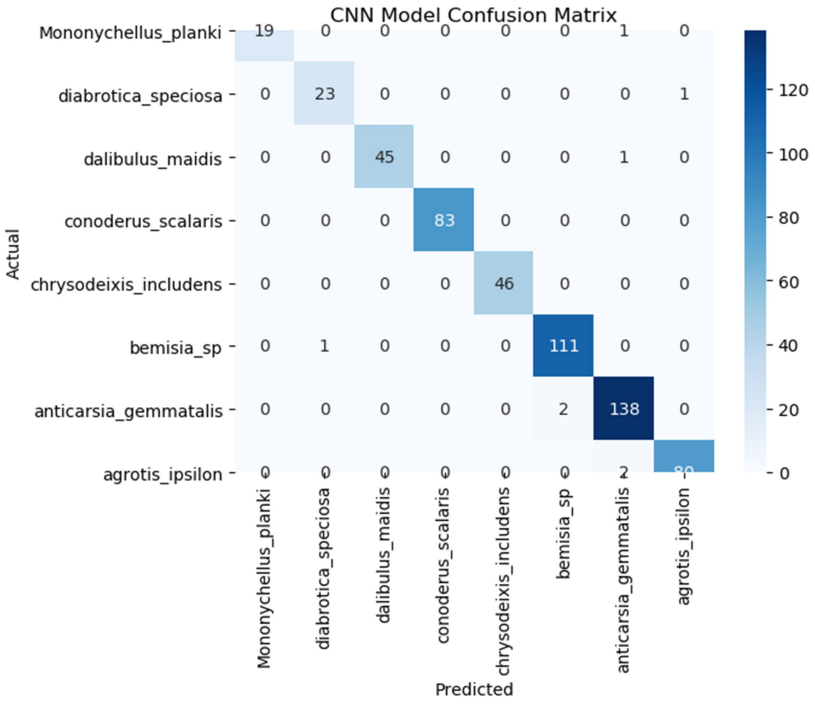

Figure 4.7 Confusion matrix of CNN model

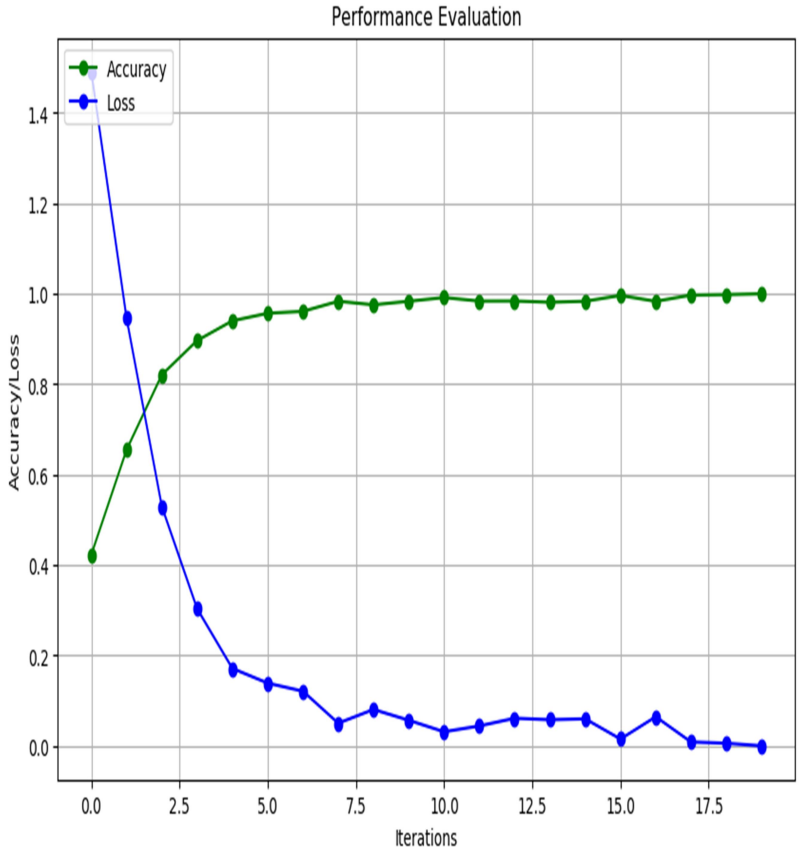

Figure 4.8 Performance Evaluation

4.3.5: uploading test image and classify

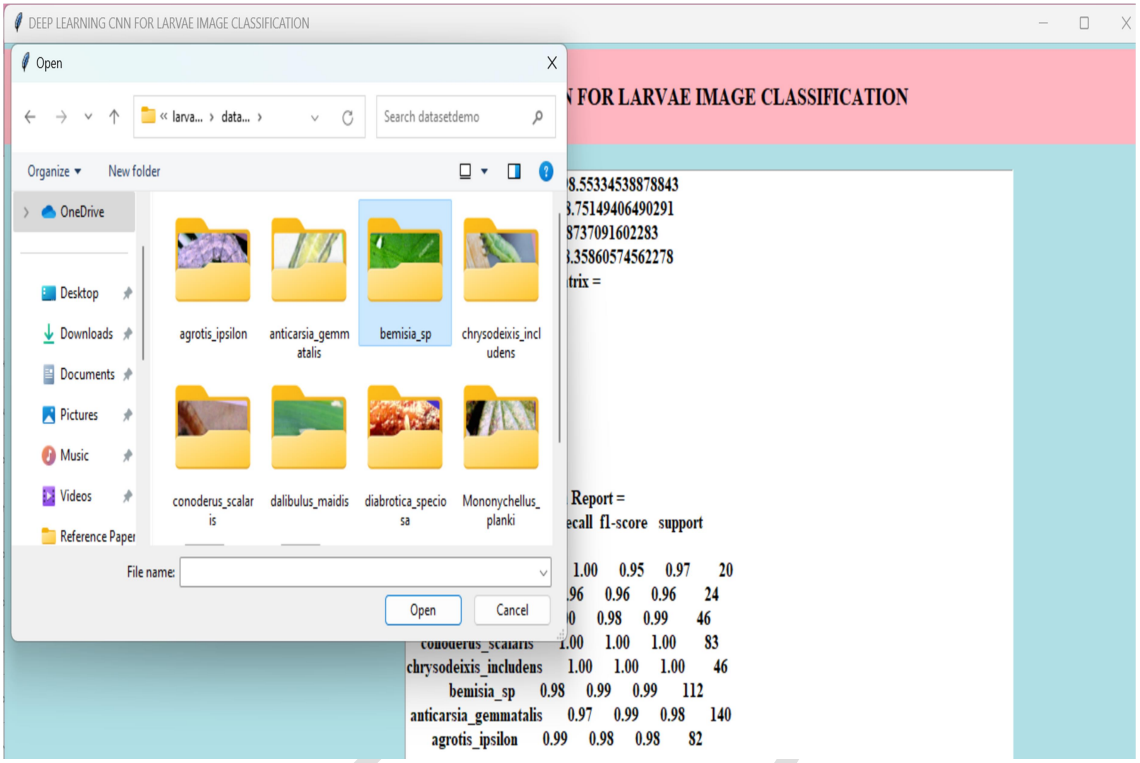

Figure 4.9 Uploading the dataset from the test data

Figure 4.10 Predicted output using Proposed CNN

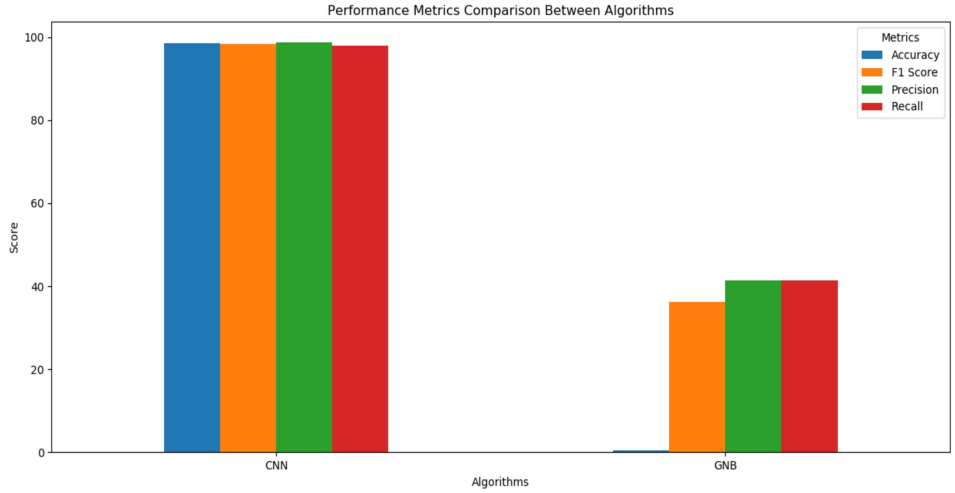

figure 4.11 Comparison of CNN and GNB Classifier

5. CONCLUSION

The implementation of Convolutional Neural Networks (CNNs) based on deep learning for larvae image classification presents a significant advancement in the field, offering a comprehensive solution to longstanding challenges associated with traditional methods. By autonomously learning discriminative features directly from larvae images, CNNs reduce the reliance on manual feature extraction and handcrafted algorithms. This not only streamlines the classification process but also enhances the model's adaptability to diverse datasets and various species, which was previously a cumbersome task requiring expert knowledge. Through extensive training on labeled data, the CNN model achieves proficiency in accurate and efficient larvae image classification, outperforming traditional techniques, especially in handling variations in lighting, scale, and orientation. This improved accuracy and efficiency make the proposed deep learning CNN architecture a valuable tool for researchers and practitioners across multiple domains, including biology, ecology, and agriculture.The scalability and adaptability of the CNN-based approach further contribute to its utility, allowing it to be readily applied to different larvae species and datasets without significant modifications. This adaptability fosters advancements in larvae-related studies and applications by providing a robust and reliable method for image classification tasks.

REFERENCES

[1] Saeed, Meer Shadman, Syeda Fahima Nazreen, Syed Shah Sufi Azmat Ullah, Zannatul Ferdaus Rinku, and Md Abdur Rahman. "Detection of Mosquito Larvae Using Convolutional Neural Network." In *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pp. 478-482. IEEE, 2021

[2] Hong, Eungi, and Mirdhini Shri Rajaram. "Identification of Mosquito Larvae in Drains Using Deep Learning." In *IRC-SET 2022: Proceedings of the 8th IRC Conference on Science, Engineering and Technology, August 2022, Singapore*, pp. 181-194. Singapore: Springer Nature Singapore, 2023

[3] Martins, Ramon Mayor, Bruno Manarin Espíndola, Pedro Philippi Araujo, Christiane Gresse von Wangenheim, Carlos José de Carvalho Pinto, and Gisele Caminha. "Development of a Deep Learning

Model for the Classification of Mosquito Larvae Images." In *Brazilian Conference on Intelligent Systems*, pp. 129-145. Cham: Springer Nature Switzerland, 2023.

[4] Lee, Sangjun, Hangi Kim, and Byoung-Kwan Cho. "Deep Learning-Based Image Classification for Major Mosquito Species Inhabiting Korea." *Insects* 14, no. 6 (2023): 526.

[5] Defèr, Adrian. "Classification of Honeybee Larval Stages Using CNNs Applied to Image Data."

[6] Gunes, Huseyin, and Ahmet Gungormus. "Identification of honey bee (Apis mellifera) larvae in the hive with faster R-CNN for royal jelly production." *Journal of Apicultural Research* 61, no. 3 (2022): 338-345.

[7] Chowdhury, Shaif, and Greg Hamerly. "Recognition of Aquatic Invasive Species Larvae Using Autoencoder-Based Feature Averaging." In *International Symposium on Visual Computing*, pp. 145-161. Cham: Springer International Publishing, 2022.

[8] Đurđević, Aca, Andrew Medeiros, Vladimir Žikić, Aleksandar Milosavljević, Dimitrija Savić-Zdravković, Maja Lazarević, and Djuradj Milošević. "Mandibular shape as a proxy for the identification of functional feeding traits of midge larvae (Diptera: Chironomidae)." *Ecological Indicators* 147 (2023): 109908.

[9] Xu, Jiajun, Zelin Feng, Jian Tang, Shuhua Liu, Zhiping Ding, Jun Lyu, Qing Yao, and Baojun Yang. "Improved Random Forest for the Automatic Identification of Spodoptera frugiperda Larval Instar Stages." *Agriculture* 12, no. 11 (2022): 1919.

[10] Srinivasarao, G., Penchaliah, U., Devadasu, G. et al. Deep learning based condition monitoring of road traffic for enhanced transportation routing. J Transp Secur 17, 8 (2024). https://doi.org/10.1007/s12198-023-00271-3

[11] Bessa, Willian RB, Vinicius N. Barbosa, Danielly G. Leite, Francisco M. M. Neto, Vinicius S. Santos, Tarcio G. Silva, Glacio S. Araujo, Mario WL Moreira, and Oton C. Braga. "Image Filtering Model Based on Convolutional Neural Networks for Automatic Counting of Post-larvae in Aquaculture." In Proceedings of the 11th Euro American Conference on Telematics and Information Systems, pp. 1-7. 2022.