Classifying Insect Pests from Image Data using Deep Learning

by

Md. Raiyan Bin Mohsin
18101639
Sadia Afrin Ramisa
18101469
Mohammad Saad
14101135
Shahreen Husne Rabbani
18101134
Salwa Tamkin
18101511

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Raiyan Bin Mohsin

Md. Raiyan Bin Mohsin
18101639

Sadia Afrin Ramisa

Sadia Afrin Ramisa
18101469

Mohammad Saad

Mohammad Saad
14101135

Shahreen Husne Rabbani

Shahreen Husne Rabbani
18101134

Salwa Tamkin

Salwa Tamkin
18101511

Approval

The thesis titled "Classifying Insect Pests from Image Data using Deep Learning" submitted by

1. Md. Raiyan Bin Mohsin (18101639)
2. Sadia Afrin Ramisa (18101469)
3. Mohammad Saad (14101135)
4. Shahreen Husne Rabbani (18101134)
5. Salwa Tamkin (18101511)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 20, 2022.

Examining Committee:

Supervisor:
(Member)

Faisal Bin Ashraf
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Tanzim Reza
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The fact that insecticidal pests impair significant agricultural productivity has become one of the main challenges in agriculture. There are, nevertheless, several requirements for a high-performing automated system that can detect pest insects from vast amounts of visual data. We employed deep learning approaches to correctly identify insect species from large volumes of data in this study model and explainable AI to decide which part of the photos is used to categorize the insects from the data. We chose to deal with the large-scale IP102 dataset since we worked with a large dataset. There are almost 75,000 pictures in this collection, divided into 102 categories. We ran state-of-the-art tests on the unique IP102 data set to evaluate our proposed solution. We used five different Deep Neural Networks (DNN) models for image classification: VGG19, ResNet50, EfficientNetB5, DenseNet121, InceptionV3, and implemented the LIME-based XAI (Explainable Artificial Intelligence) framework. DenseNet121 performed best across all classes, and it was also employed to detect crop-specific insect species. The classification accuracy for eight specific crops ranged from 46.31% to 95.36%. Moreover, we have compared our prediction performance to that of earlier articles to assess the efficacy of our research.

Keywords: IP102, Insect pest, Transfer learning, Data augmentation, classification.

Dedication

We dedicate the report to our parents. Without their participation, attention, and support, we would not have gotten this far. We owe a debt of gratitude to them. Many thanks to them.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our supervisor Mr. Faisal Bin Ashraf sir and co-supervisor Mr. Md. Tanzim Reza sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$\sum$ Summation

ANN Artificial Neural Networks

CNN Convolutional Neural Networks

DNN Deep Neural Networks

EC Economic Crops

FC Food Crops

KNN K-Nearest Neighbors

MSE Square Discrepancies

NS Number of Samples

ReLU Rectified Layer Unit

SVM Support Vector Machines

TN True Negative

TP True Positive

Chapter 1

Introduction

In agriculture, insect pests are one of the most severe challenges since they wreak havoc on crop quality. Crops are afflicted by pests, viruses, resulting in food scarcity on the market. Insects devour around 10% of ordinary produce; at times, the amount of wheat, rice, and other grains consumed can be substantially more significant. It has found around 10,000 distinct parasitic bug species. Plants have been harmed over a long period. Identifying multiple variants with minor differences is difficult, especially if not all of these species are believed to be necessary. Being innovative in pest detection, even if only to improve efficiency and efficacy somewhat, might be the difference between profit and loss. Agriculture requires continuous monitoring and evaluation of insect losses to ensure the quality and safety of crops. Due to the wide variety of pest species and their subtle distinctions, insect pest identification primarily relies on agricultural professionals' professional experience, making it more expensive and time-consuming. However, while human interaction is required, a machine-based automation solution can quickly relieve the earlier issues. Automated insect pest monitoring is becoming more familiar with developments in machine learning and computer vision techniques.
This research aims to classify and detect insects in fields and develop a model capable of accurately identifying nuisance insects from images. The insect pest identification method for agricultural insect collection is practical and accessible in terms of calculation time. We utilize the IP102 dataset [20] to further insect pest recognition studies in computer vision. Around 75,000 images of 102 different species are included in the IP102 dataset [20]. This research illustrates and analyzes 102 different insect pests types and ways to detect and categorize insect pests from images. We implemented VGG19 [4], ResNet50 [6], EfficientNetB5 [18], DenseNet121 [9], and InceptionV3 [7], all of which achieve high levels of accuracy when presented utilizing the LIME-based XAI system. This study compared the performance of five unique designs.

1.1 Research Problem

Agricultural companies and farms emerge day after day where technology plays a crucial role. But the progress of agriculture continues to be significantly impeded by insects/pests. Although we can control different chemical and biological pesticides, careful monitoring is typically necessary across the site to get the most out of them. In many cases, workers do passive surveillance while performing daily activi-

ties. Therefore, frequent human monitoring cannot adequately predict the number and intensity of pests or diseases attacked on the farm for spraying the right fertilizers/pesticides to remove the host. Problem identifying is excellent, as a system such as this must differentiate between the intended species and cope with many non-target species. This makes complicated machine learning approaches more common than detection. There were numerous methods for detecting insects/pests. For instance: Visual inspection, Trap method, PIR motion detectors, sensors (Thermal sensor, Fluorescence Image Sensing, Acoustic Sensors, Gas Sensors, Sound wave sensor, i.e.), ANN classifier, i.e., Researchers later implemented more complex version classifiers, such as the KNN classifier, SVM classifier, and CNN classifier, as technology progressed. In recent years, we can observe that though there are so many ways for detection and identification, researchers cannot find a form with higher accuracy. In addition, many approaches are time-consuming and expensive. As a result, the farming industry suffers alarmingly from insects/pests.

1.2 Research Objectives

Our initiative aspires to more effective detection and identification by Image Processing of insects/pests and disease assaults. This allows us to detect insects from vast image data using deep learning. The following are the key aims of our research:

- Insects/pest detection and identification through Image Processing
- Detecting and identifying with a comparatively higher accuracy rate
- Method effectiveness test using advanced algorithms
- Efficient process to safeguard and sustain the environment
- Adaptability and affordability

There may be other choices, but additional research is needed.

1.3 Thesis Structure

In chapter 1, we discussed the Research problem and research objectives. In chapter 2, we discussed Literature review and deep learning. In chapter 3, we discussed Methodology and proposed models. In chapter 4, we discussed Dataset and data augmentation. In chapter 5, we discussed Implementation and results. In chapter 6, we discussed the Conclusion and future work.

Chapter 2

Background Study

2.1 Literature Review

Pest insects have long-term detrimental effects on our agricultural, natural, and lifestyle ecosystems. Insects from pesticides can damage crops and food production and even produce human health risks. It is used in the core domains of "deep learning" in computer science.
Like the human brain, deep learning is one of the domains of algorithmic machine learning (Artificial Neural Networks).
But when it comes to insects - since it is so tiny, it is color camouflage with nature or sometimes with plants - so we have several dimensional issues. DEEP CNN has been used to identify small things in agriculture in many methods. Image solutions are various and have previously been wholly explored to identify species [29]. The hyperplanes are used to identify insects, which separate the classes.
The main aim of the convolution layer, which works as filters, is to extract characteristics from bug photographs. The data they split for classification using deep neural networks as classification devices was used to detect and identify indoor and outdoor food congestion using a single shot multi-box detector (SSD) with a clustering network based on VGG16. In general, the results of this study are mixed (45% MAP), but the results of herbivore classification in the 8-class research in 1105 test samples are objectively optimistic, with 81.5% of their accuracy. Some further instances used the Wang and XIE datasets to use their models with 24 different insect species. For the Vector Machine (SVM), closest adjacent to KNN, naïve Bayes (NB), certain forms and approaches are used (CNN). A high-resolution model was developed for pest surveillance. It was built using a deep residual recursive network. With 91.5% for the nine courses and 90% for the 24 classes of Wang and Xie, the CNN model obtained the most remarkable accuracy in the research. After extensive investigation, we reported on both a small dataset and the more extensive IP102 dataset. They analyzed their methods. They have achieved a state-of-the-art accuracy on both smaller datasets (92.43%) and bigger (61.93%), corresponding to the performance of human experts. They have proposed a new procedure for classifying insect pests using coevolutionary neural networks (CNNs). The most important feature of a picture is emphasized via image processing techniques known as saliency methods (Itti, Koch, and Niebur, 1998). Our research seeks to help farmers safeguard their fields from attacks by pests and diseases while preserving the integrity of the soil and other plant regions. This means that Deep Learning is used to identify insect

pests from extensive picture data.
This section primarily focuses on the vital work in deep training in insect and pest identification in agriculture. The primary goal of the research is to use technology to increase the production of crops. Start-ups and technology firms apply artificial intelligence, machine learning, deep learning, and IoT technologies to boost agricultural output. We have built many smartphone apps to detect the weather, soil, and crop species we can exploit for the maximum yield feasible.
When 'target' and 'objective missing' classes are present, identifying and categorizing pests over the complete scenario is vital for binary classification.
On different field plants for insect categorization and insect detection, Wang and Xie's datasets have been finished [6]. Wang uses a set of nine insect classes with 225 images, whereas Xie uses a single set of 24 classes with a photograph of 25 per class and a ratio of train testing between 70% and 30%.
The training set consists of 162 images of insects in the data set of Wang and 63 images from insects in the test set. Xie Data Sets includes 785 insect photographs and 612 insect photos in a test set consisting of roughly 60 bug photographs each class [6].
Grain safety research in grain storage highlights the detection and identification of grain insects that deliver greater yields [18]. The saved grain insect detection and identification database are not typical. ASAG China collected eight species from stored grain insects (State Grain Administration Academy) [18]. Over 1,000 images and 70% training have been randomly picked and 30% training. We have built a second data set to replicate the present green warehouse condition with a total resolution of 784 original photographs from 2592 to 1944. Insects were around eight pictures, averaging their density [18].
Research shows how deep learning algorithms examine identification and prediction to acquire the required database. They are part of it. Use MATLAB, the Neural Network Toolkit, the Bioinformatics Toolbox, and the Image Processing Toolbox to address this challenge. After interviewing different farmers, their survey indicated that most farmers want to receive applications online 24×7, for free. The farmer has to do nothing about it; he only needs a picnic and a cloud upload. The back-end processing makes the real quick test and offers the farmer a full overview report.
Wang, Xie, Dang, and IP102 were used to research a highly complex environment. Firstly, the insects were categorized using image processing methods, by which the image noise is minimized and images improved to higher accuracy. This procedure employs an algorithm that retrieves the environment and the detection of texture. The photos of the pests were magnified to a higher resolution to reduce the detection uncertainty.
An additional study employing neural system arbitration produces an adaptive system for identifying the insects, categorizing eight of the insects seen routinely in rice fields primarily completed in two steps. The photographs will mainly be processed and their patterns identified by many images processing tools. The segmentation approach is used to level the data and then separate and prepare neural networks' backpropagation for a neural network. Then, two training algorithms in the input portion are developed using various learning techniques during the identification stage. They are then tested on many raw photos until they are selected. The texture of each insect is preserved.
The pest identification research utilizing machine learning classifies the picture by

taking the picture raw file and transforming this picture into precious images for a specific test [29]. The photo is taken at high resolution and subsequently decreased using a quantified image representation, which keeps vital classification issue information. The result is produced as a vector. Machine education is then utilized to recognize existence. Afterward, it is comparable to other scores of any other living objects such as plants and animals to identify the species [29].
The research utilizing deep learning to detect and predict pests and diseases focuses on the farming industry's economic sector. The artificial neural network is applied to study the condition of the plant and soil. The dataset of these plants and soil is frequently updated in the MATLAB tool, which helps identify. The Machine Learning Algorithm then uses CART (classification and regression tree) to forecast future pest or disease attacks.
This study is based on a long-term relationship between insects and plants and employs the herbarium exemplifier. Machine learning assumes that just a minimal amount related to identifying processes are based on these facts. Data from the South East's regional expertise and collections network, Q. bicolor, and O. sensibilis are downloaded. The detection and classification studies were therefore separated into two groups and each independently simplified.

2.2 Deep Learning

The characteristics retrieved from the image pertain to the "graphics" of the model, and the object's choice is quite problematic. It was essential in classifying performance in the past, but it was also labor demanding and subjective expert work that manually derived characteristics. In addition, we cannot manually extract many of the features accurately. Consequently, a technique that would automatically identify appropriate functionality for a problem with a defined logic endeavored after.
Deep learning is an AI characteristic that replicates the human brain's data-processing function for object recognition. Artificial neural networks automatically extract data from collected samples by learning the proper representation and applying a solid model. This automatic removal proof is accurate to computer vision, cutting-edge image technology, recognition of objects, and image recall models. (Bengio, Courville, & Vincent, 2013) [12]. Deep learning is based on widely used ANNs that apply mathematical models in the biological, neural, centralized animal nervous system and brain-inspired learning algorithms. Neuronal networks consist of one or more deep neuronal learning layers, which combine the bulk of the Artificial Brain, composed of numerous hidden layers. ANN expanded more extensively via the flow of information and dispersed biological communication nodes but still varies in several ways from the human brain. The term "deep" is used here to signify that this network contains more than one layer. Initiators and tech businesses utilize profound learning and IoT technology to enhance agricultural yield [20]. Deep learning makes use of Neural Networks to understand how the human brain operates.

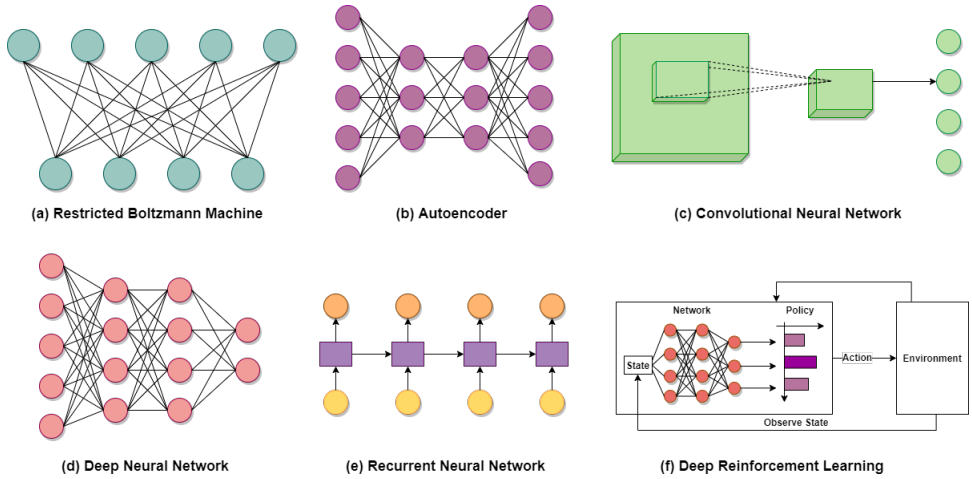

Figure 2.1: The structures of different deep learning models.

These networks are like distinct nodes (or places) linked in a single layer in the human brain. The more the number of layers, the stronger the network gets. Information travels between nodes using signals. The relevant weights are applied to the nodes when these signals are received. Heavier-weight knots will increase the impact on the other neighboring nodes. The weighted inputs are transformed to outputs afterward. The entire system required costly hardware since vast data, including multiple complex computations, had to be processed.
The complete procedure may be conducted using the image and processing description, object sensing network, and model optimization for insect and pesticide identification.

2.3 Convolutional Neural Networks

Convolutional neural networks and CNN are among the most common deep learning algorithms used to recognize patterns and classify images. CNN algorithm consists of several contemporary designs. In Keiron O'Shea's opinion, Ryan Nash, CNNs comprised neurons that may be optimized to accept feedback and complete a task wherever each neuron continues. CNN is composed of three-dimensional neurons: the spatial dimension of the input (high and broad) and the depth. Three layers, involute layer, pooling layer, and entirely linked layers, form CNN architecture. The convolutional layer defines the output of neurons connected to local areas input to compute the scalar-produced product between its weights and the region associated with input volume. An image transformed into a vector goes through a two-dimensional weight set, kernel, or filter. The input data and kernel are used for a dot product.

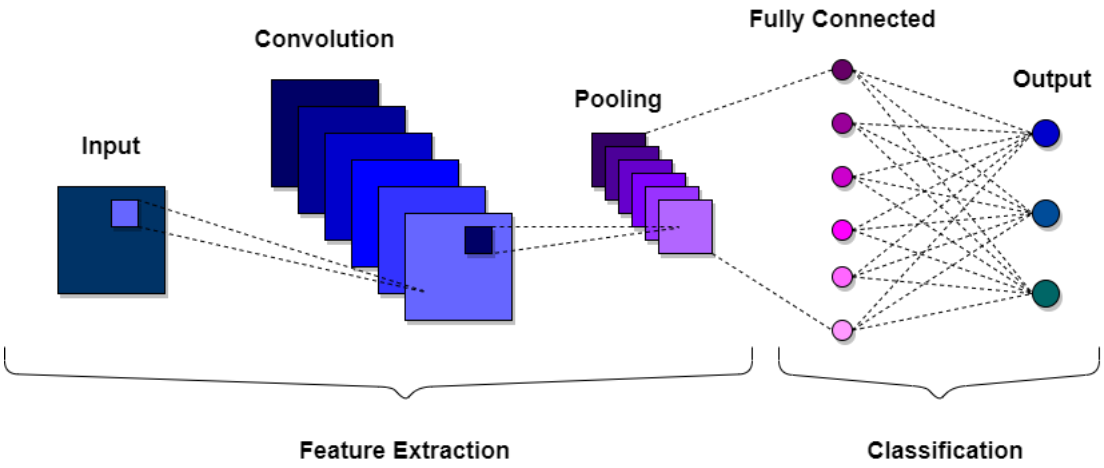

Figure 2.2: Convolutional neural network architecture.

Once the kernel has been applied systematically over the picture, a two-dimensional array called the feature maps is produced. Activation functions such as ReLU are used to "excite" the kernel when a specific characteristic occurs at a particular input spatial point. The activation map is transmitted over the pooling layer after passing through the convolutional layer. Pooling aims to reduce the architecture's dimensional and complexity. Ultimately linked layers include neurons connected directly in the two neighboring layers, without any layers connected beyond them.

2.4 Explainable Artificial Intelligence (XAI)

XAI which is used to denote Explainable Artificial Intelligence is several processes and techniques, used by different types of algorithms, bring a perfect output that can be understood by humans. Any complex model or system which are studied nowadays can easily be presented and explained with the help of Explainable AI. It can be used to build a strong association between the service providers and consumers. Furthermore, this will also assist in any future works for further development.
We often work with algorithms that train themselves with the help of a training dataset that is curated with different sizes of datasets and makes decisions by itself. It becomes difficult for us to understand on a deeper level. The main arena of Neural Networks and Artificial Intelligence is the mighty concept of Black Box which helps to understand the scenario perfectly. Data experts and Scientists sometimes find it difficult to understand how the final output is achieved because the process undergoes a series of steps. To understand the entire event knowledge graph is used which helps us with a clear view of the entire procedure.
System developers often expect a detailed view of the entire process of how the out is generated, as a lot of important factors are involved behind it. We often find it hard to completely figure out the sequence of steps involved in neural networks, thus it is labeled as black box which is generating the final output.

Chapter 3

Methodology

The suggested approach involves dataset acquisition, DNN model training, test picture classification, and final findings. We had to divide up the dataset to train and evaluate the models.
We maintained the training, validation, and testing sets similar to those stated in the research to ensure appropriate comparability with the original study. To generate the training, validation, and testing sets, the dataset was divided about 6:1:3. As a result, 45,095 pictures were assigned to the training set, 7,508 to the validation set, and the remaining 22,619 as the testing set. We required many pictures to feed the big DNNs, and thus, the training set was the largest.

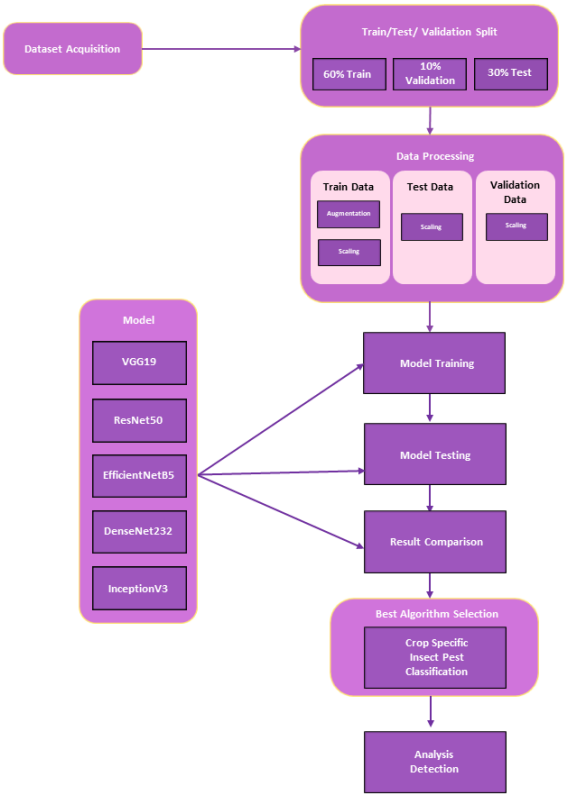

Figure 3.1: Proposed method of the research.

3.1 Implemented Models

Deep learning algorithms are used to study, retrieve and accurately classify the training set from the test system, unseen data. We also use transfer learning to decrease the training period of the system and the limited data we have. Numerous deep learning algorithms and models have evolved due to increased research in deep learning and image categorization. Our dataset examined the performance of five architectures: VGG19, ResNet50, EfficientNetB5, DenseNet121, and InceptionV3. We attempted to compare the results. The models we utilized are detailed further down.

3.1.1 VGG19 [4]

For Visual Geometry, Group VGG is an abbreviation (a group of researchers at Oxford who developed this architecture). The VGG architecture consists of 2D Convolution and Max Pooling layers. VGG uses 1×1 convolutional layers to reduce the decision function while maintaining the same receiving fields. Due to the small convolution filters, VGG may have several weight layers; more layers mean, of course, improved performance. The initial releases in 2014 were VGG 16 and VGG 19. The amount of weight layers is at the end of VGG19.
VGG-19 was formed on millions of photos in the ImageNet collection by a convolutional neural network. The network can divide photos into 1,000 types of objects, such as keys, mice, pencils, and animals. The network has so collected a large number of features for several images.

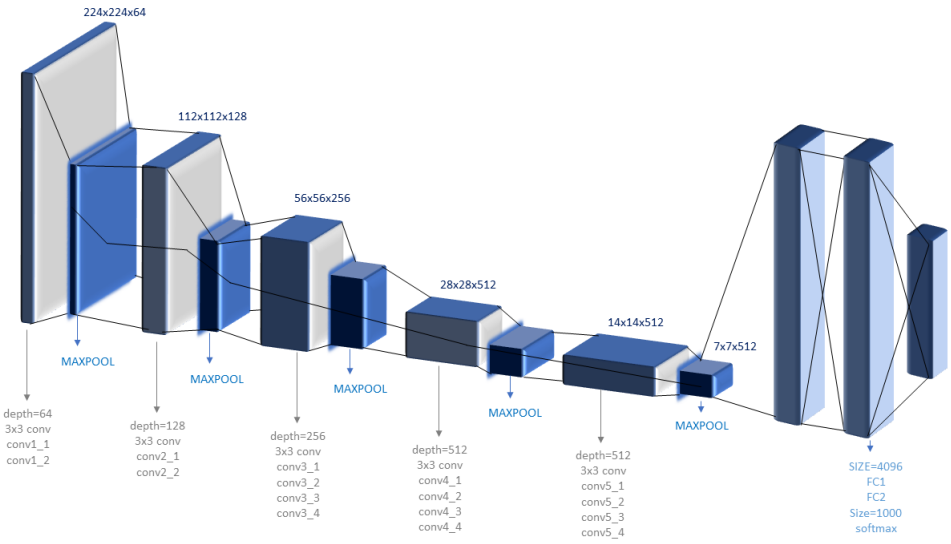

Figure 3.2: VGG19 convolution layers.

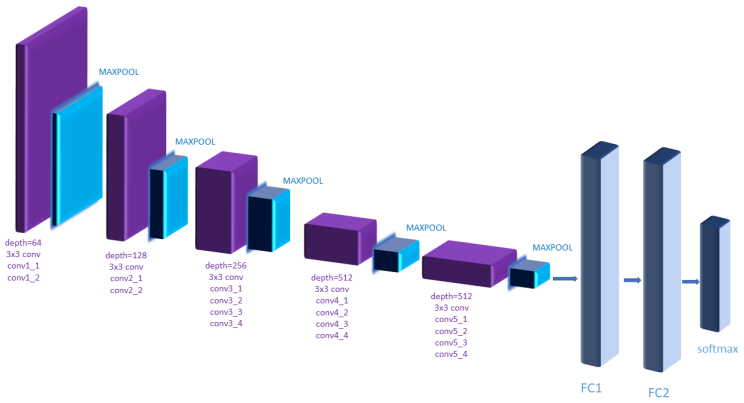

Figure 3.3: VGG19 network architecture.

The model VGG19 is a version of the 19-layer VGG model (16 layers of convolution, three levels of ultimately linked layers, five layers of MaxPool, and one layer of SoftMax). The input to the layer cov1 is a fixed-size 224 to 224 RGB image, which is then processed using a stack of convolutional layers (Convolutional) with a very narrow receptive field: 3 to 3 (the smallest size in which left/right, down, and center) ideas are captured. It also uses a T1 convolution filter, a sequential change of the channels input (after nonlinearity) in one setup. The convolution stage is single-pixel wide. The spatial resolution is reserved, which is done after the convolution, where it is done in a way where a single pixel is set for three convolutional layers. Five layers of maximum pooling, after numerous layers of Convolutional (max-pooling doesn't follow all the layers). To pool max in step 2, a 22-pixel frame is utilized. After a sum of convolutional layers, three fully connected (FC) layers were added (various depths in different designs). The first two have a total of 4096 channels. The following classification is 1000-way ILSVRC, which operates concurrently on 1000 channels (one for each class). The last layer is a soft-max one. Every network has the same ultimate connected layer configuration. The nonlinearity of the correction (ReLU) is available on all burial strata. It should also be emphasized that, except for one network, no Local Response (LRN) function will be used to increase memory consumption and computational time rather than improve the performance of the ILSVRC dataset.
VGG offers advantages: the architecture for evaluating individual jobs is excellent. Pretrained VGG networks are likewise free of charge available on the internet and are thus commonly used outside the box for numerous applications. The CNN model was mainly aimed towards winning ILSVRC. But in many different ways, it has been used.

- The model may be used for many additional datasets as a good classification architecture. The authors have made the models accessible to the public, as it is or with modifications in other comparable jobs.
- Transfer learning: may also be utilized to recognize face tasks.
- With other frameworks like Keras, weights are simply available to be used as one desire.
- Loss of content and style via the network of VGG-19.

3.1.2 ResNet50 [6]

ResNet is a deep convolutional neural network usually coupled with image recognition, classification, and auto-encoding. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was won using function transmission to avoid gradient loss and quickly training a network far more profound than previously employed.

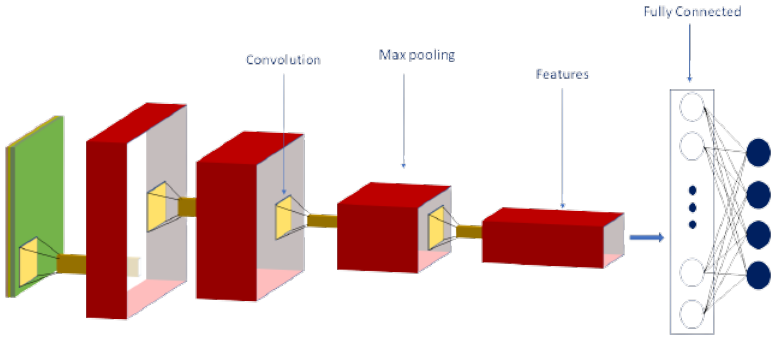

Figure 3.4: ResNet layers.

Driven by earlier research, which shows that a closer network is possibly more potent than external networks, the remaining networks were built up from a 50-layer residual network design, and have a total of 177 layers, showing detailed information in the Supplementary Information (ResNet50). In the ImageNet database (https://image-net.org/), ResNet50 was trained to classify pictures into 1000 categories of objects (e.g., keyboard, mouse, and pencil, as well as several animals and insects).

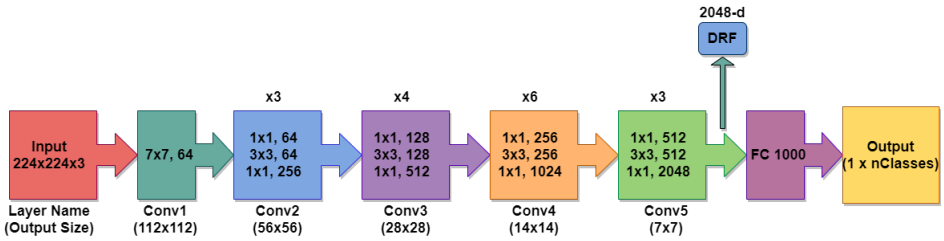

Figure 3.5: ResNet50 architecture.

The residual units, the filter's size, and the output of each convolutional layer are all specified as shown in the architecture of ResNet-50. DRF is also exhibited from this network's final convolutional layer. Key: The notification "k" means a filter with the size "k" and "n" in the convolutional layer block. FC 1000 is the layer of 1000 neurons wholly linked. This number is the repetition in each unit at the top of the convolutions of the layer block. nClasses denote the number of output classes.

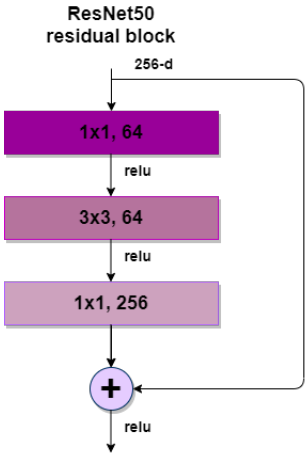

Figure 3.6: Residual block.

The ResNet50 architecture comprises the following components, as we can see in Figure:

- A convolution of 7 ×7 and 64 distinct kernels with a stride of size 2 results in the formation of a single layer.
- We observe the subsequent max size pooling with a size 2 as well.
- The subsequent convolution consists of one kernel (1×1,64), followed by one kernel (3×3,64) and a 1×1,256 kernel repeated for 3 layers so that 9 layers are specified in this stage.

Table I

ResNet Layers

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2.x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$
conv3.x	28×28	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times8$
conv4.x	14×14	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times36$
conv5.x	7×7	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8\text{x}10^9$	$3.6\text{x}10^9$	$3.8\text{x}10^9$	$7.6\text{x}10^9$	$11.3\text{x}10^9$

Table 3.1: Layers of ResNet

We exclude the activation functions and the intermediary layers used for pooling. This results in a Deep Convolutional Network with a total layer count of $1 + 9 + 12 + 18 + 9 + 1 = 50$ layers.
While ResNet is straightforward to optimize, “plain” networks (those that stack layers) exhibit a considerable rise in training error as the depth of the network grows.

ResNet may quickly increase accuracy as it gains depth, resulting in superior outcomes than previous networks. Without expanding the training error %, it is possible to train networks with numerous layers (even thousands). ResNet may help with the vanishing gradient issue by using identity mapping. ResNet significantly improves the performance of neural networks with additional layers. The Deep Residual Network features a bottleneck residual block design to enhance network performance. This method may be used for picture categorization, object localization, and object detection in computer vision. It may also profit from depth in non-computer vision applications while decreasing processing costs. ResNet is predicted to outperform standard deep neural networks, if not surpass them. On the other hand, ResNet outperforms by a huge margin when the network gets deeper. ResNet's skip connections allow the gradient to flow down an additional shortcut channel, preventing the slope from vanishing in deep neural networks. Additionally, these linkages aid the model in learning identity functions, ensuring that the top layer performs well while the bottom layer performs poorly.

3.1.3 EfficientNetB5 [18]

EfficientNet is simply a convolutional neural network design and a scaling strategy that applies a compound coefficient to the depth/within/resolution dimensions. Google AI [18] has released EfficientNet, which may enhance the performance of image classification compared with cutting-edge efforts. EfficientNet scaling can jointly improve the properties of CNN, e.g., its breadth, depth, and resolution. So far, a total of 7 EfficientNet, EfficientNet-B0, and B7 versions have been launched. The number of layers utilized is the distinctive aspect of these versions. For example, in EfficientNet-B0 and EfficientNetB7, the coatings used are 237 and 813, respectively. We use a picture size (456, 456, 3), EfficientNet-B5. This suggests that 456, 456, and 3 channels are the picture height, breadth, and channels. The size of the batch utilized is 4. We can readily learn from Figures 3.7, 3.8, and 3.9 that many layers are connected in EfficientNet-B5. The stem and the last layers of the network are shown in fig. 3.7. The module set that is connected and repeated is presented in Fig. 3.8. In figures 3, $\times 3$, $\times 5$, and $\times 7$, the modules inside the brackets are repeated three, five, and seven times. EfficientNet-B5 can produce considerably better results than EfficientNet-B0 to B4 when utilizing the same dataset.

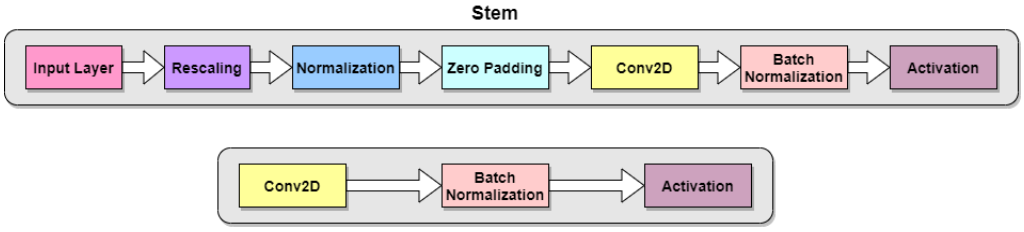

Figure 3.7: Stem and final layers of EfficientNet-B5.

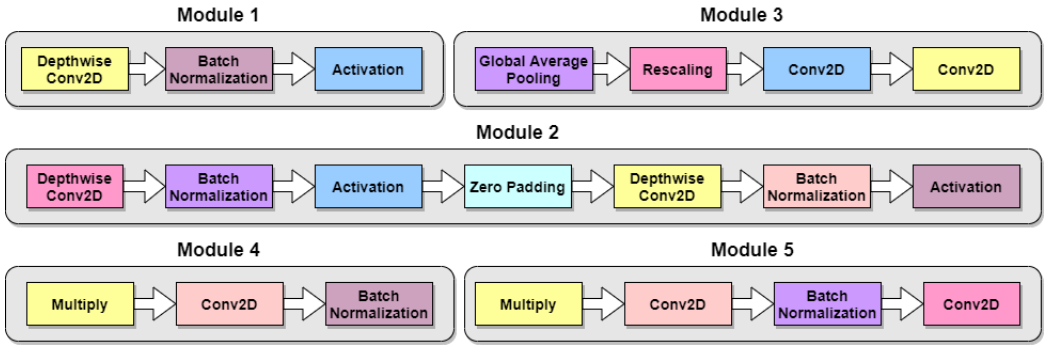

Figure 3.8: Modules of EfficientNet-B5.

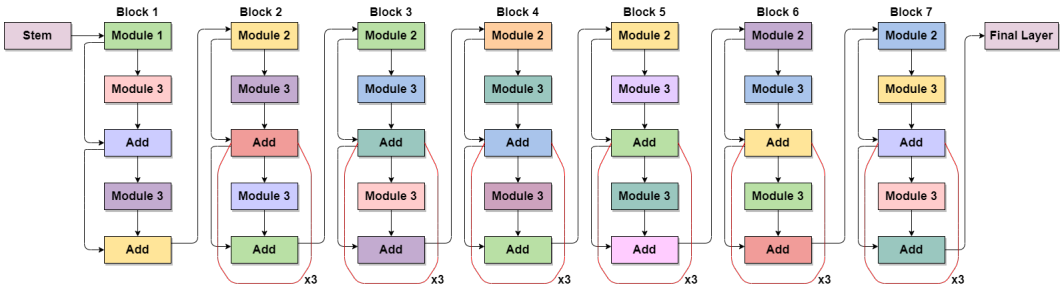

Figure 3.9: The total architecture of EfficientNet-B5.

Since we optimized the QWK score and MSE, we formulated the problem as a regression problem. This method allows us more freedom in optimization, which means that we can get more outstanding QWK accuracy scores and lower MSE loss values. The EfficientNet-B5 with additional layers has been pre-trained. MSE is the square discrepancies between our anticipated values and those observed as follows:

$$MSE = \frac{\sum_{i=1}^{n}(\widehat{y_l} - y_i)^2}{n} \tag{3.1}$$

Where, y_i = observed values, y_i = predicted values, n = number of data points
In addition, we have increased the data to improve the model's robustness. The data was rotated and turned vertically and horizontally. To improve convergence, we employed group standards and Adam optimizer. Finally, we split the values by 128 to standardize them. We educated all network layers in this way. EfficientNet uses a compound coefficient to equally and reliably measure network dimensions and resolution. A predetermined set of coefficients for scaling, randomly modifying these parameters, is used to increase network width, depth, and resolution reliably. Suppose, for example, we want to use 2N times more computer resources. In such cases, the network depth can be increased by αN, the width by β N and the image by β N; α, β and α are constant coefficients derived by a grid search in the tiny initial model.
The compound escalation method is based on the idea that more layers and channels are necessary to increase the receptive field and get more fine-grained patterns on the larger image as the picture gets more extensive. On multi-class image classification on data sets, EfficientNet will be implemented. It is better to use the EfficientNet-B5 version because B6 and B7 do not support Keras weights for ImageNet.
The critical contribution of EfficientNet was to study in-depth how to measure the

size of the convolutional neural networks efficiently. Examples include the size of the ConvNet by using the layer width, the layer depth, image input resolution, or a combination of all three levers.

3.1.4 DenseNet121 [9]

The DenseNet121 model is one of the image classification models in the DenseNet collection. The authors trained the models using Torch before converting them to Caffe format. All DenseNet models were prepared using the ImageNet image database.
DenseNet begins primarily with a fundamental level of convolution and pooling and a sequence of transition layer and dense blocks, and in the end, there is a classification layer. The initial convolutional block has 64 filters with a size 7×7 and a stride of 2. A max Pooling Layer follows it with the same stride and a 3×3 max pooling.

Table II

DenseNet Layers

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolutional	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times6$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times6$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times6$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times6$
Transition Layer (1)	56×56 28×28	1×1 conv 2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times12$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times12$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times12$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times12$
Transition Layer (2)	28×28 14×14	1×1 conv 2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times24$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times32$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times48$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times64$
Transition Layer (3)	14×14 7×7	1×1 conv 2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times16$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times32$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times32$	$\begin{bmatrix}1\times1\\3\times3\end{bmatrix}\times48$
Classification Layer	1×1	7×7 global average pool 1000D fully-connected, softmax			

Table 3.2: Layers of DenseNet

The table above summarizes the many architectures used to build the ImageNet database. The number of pixels shifted across the input matrix is the stride. Astride of 'n' (the default is 1) specifies that the filters turn 'n' pixels every time.
To explain the table, we may use the DenseNet-121 design, which shows that each dense block contains a changing number of layers (variations) with two convolutions apiece; an 11-dimensional kernel for the bottleneck layer a 33-dimensional kernel for the convolution procedure.
Additionally, each transition layer has an 11-layer convolutional layer and a 22-layer average pooling layer with two strides. As a consequence, the following layers are present:

- A basic convolution layer comprised of 64 filters with a size of 7×7 and a stride of 2.

- A layer of pooling with a maximum pooling of 3×3 and a stride of 2.
- Dense Block 1 is composed of two convolutions repeated six times.
- Layer 1 of the transition (1 Convolution + 1 AvgPool).
- Dense Block 2 has 12 repetitions of 2 convolutions.
- Layer 2 transition (1 Convolution + 1 AvgPool).
- Dense Block 3 consists of 2 convolutions that are repeated 24 times.
- Layer 3 of the transition (1 Convolution + 1 AvgPool).
- Dense Block 4 is composed of 16 convolutions.
- To execute classification at the Output layer, the Global Average Pooling layer accepts all of the network's feature mappings.
- Output layer.

As a result, DenseNet-121 is composed of the following layers:

- 1 7×7 Convolution.
- 58 3×3 Convolution.
- 61 1×1 Convolution.
- 4 AvgPool.
- 1 Fully Connected Layer.

DenseNet-121, in summary, has 120 Convolutions and 4 AvgPool.
Layers, even within the similar dense block and transition layers, distribute their weights on several inputs, that enables more deep layers to exploit attributes gathered earlier in the process.

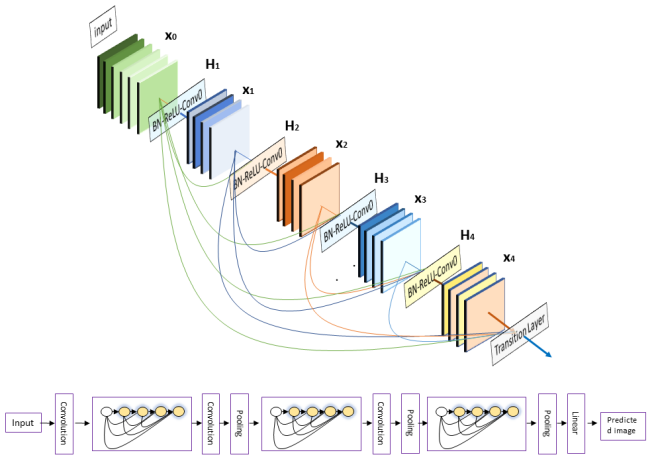

Figure 3.10: Complete architecture of DenseNet121.

Minimum weight is given by the second and third blocks to the transition layers, producing several repeated features. Additionally, even though the final layers use the consequences of the whole dense block, more superior features formed more profound in the model since there seemed to be a more excellent concentrate on final feature maps throughout trials.
Because DenseNets need fewer parameters and enable feature reuse, they build more compact models and attain state-of-the-art performance and outcomes across competitive datasets than their regular CNN or ResNet counterparts.
Thus, DenseNets provide numerous appealing benefits, including removing vanishing-gradient issues, enhanced feature propagation, feature reuse, and a significant decrease in parameter count. It creates a strong gradient flow, and the error signal may be sent more directly to the preceding tiers. This is a kind of implicit deep supervision since previous levels might get direct control from the final classification layer.
DenseNet provides a greater variety of features and patterns since each layer takes input from all preceding levels. It retains low complexity features since DenseNet's classifier utilizes characteristics of all complexity levels. It often establishes more nimble decision-making limits. This also explains why DenseNet operates well without enough training data. Additionally, it is more efficient in terms of parameterization and calculation. As with DenseNet, the network now has '(I (I + 1)) / 2' connections, similar to conventional deep learning architectures. Consequently, it needs fewer parameters than typical convolutional neural networks, as it does not require the learning of useless feature maps.
DenseNet is a network design that emphasizes incrementing deep learning networks and increasing their training efficiency using the minimum connection between layers. DenseNet is a convolutional neural network in which each layer is linked to all subsequent layers, where each layer is connected to all the alternate layers following it. This guarantees that the network's tiers can exchange the most significant amount of data possible. The feature maps are transmitted to all future layers from the preceding levels of each layer, and thus the systems feed-forward nature is preserved.
The objective of developing superior higher-layer designs resulted in the invention of this architecture. Specifically, addressing the problem of several levels becoming redundant in multilayer networks. The DenseNet plan aims to address this issue by tightly connecting all tiers. This implies that each layer inputs from all preceding levels and sends information to all subsequent layers. The consequence is that the highest output layer has direct access to all initial layers, including the first. This is meant to aid in resolving the duplicate layer problem.

3.1.5 InceptionV3 [7]

InceptionV3 is a convolutional neural network module created by Google for image processing and object detection. It was designed in such a way so that it can dig through deeper networks while maintaining the parameters. Convolutions, average pooling, maximum pooling, concatenation, dropouts, and ultimately connected layers are all the model's symmetric and asymmetric building blocks. Batch norm is used frequently in the entire model to determine activation inputs, while Softmax calculates the loss.

The InceptionV3 model has 42 layers, somewhat more than the InceptionV1 and V2 models. However, the effectiveness of this approach is astonishing. We'll get to it shortly, but first, let's examine the components that comprise the InceptionV3 model.

Table III

InceptionV3 Architecture

Type	Patch/Stride Size	Input Size
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv Padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3×Inception	Module 1	35×35×288
5×Inception	Module 2	17×17×768
2×Inception	Module 3	8×8×1280
Pool	8×8	8×8×2048
Linear	Logits	1×1×2048
Softmax	Classifier	1×1×1000

Table 3.3: The architecture of InceptionV3.

The architecture of an InceptionV3 network is progressively built, step-by-step, as explained below:

- **Factorized Convolutions:** The reduction of the number of parameters in a network contributes to its computational efficiency. Additionally, it keeps an eye on the network's efficiency.
- **Smaller convolutions:** Substituting smaller convolutions for bigger convolutions significantly accelerates training. A 5×5 filter, for example, has 25 parameters; two 3×3 filters, used in 5×5 convolution, have just 18 parameters (3×3 + 3×3).

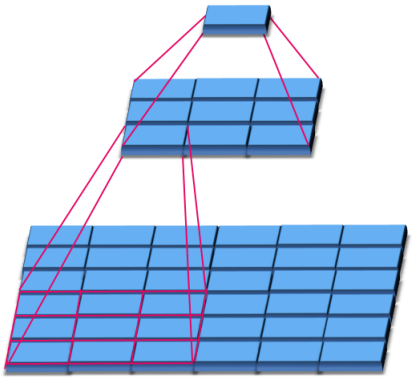

Figure 3.11: Breaking into smaller convolutions.

In the middle we see a 3 ×3 convolution, and below a fully-connected layer. Since both 3 ×3 convolutions can share weights among themselves, the number of computations can be reduced.

- **Asymmetric convolutions:** A (3×3) convolution might be substituted with a (1×3) convolution followed by a (3×1). If a (3×3) convolution is substituted for a (2×2) convolution, the number of parameters is significantly more than with the suggested asymmetric convolution.

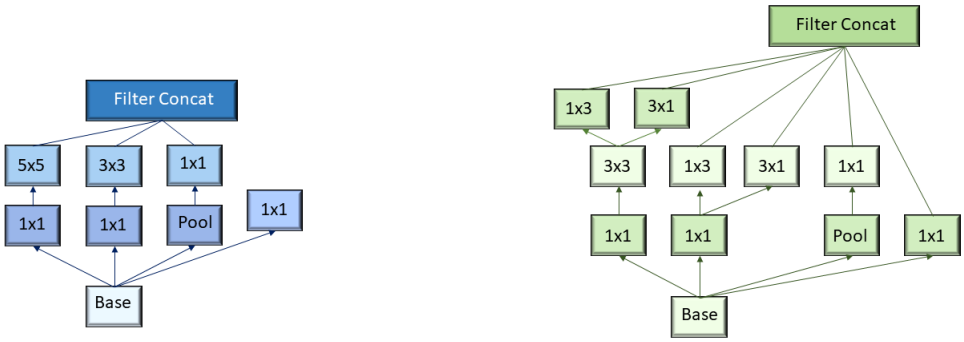

Figure 3.12: Asymmetric convolutions.

- **Auxiliary classifier:** During training, a smaller CNN is put between layers as an auxiliary classifier, and its loss is added to the main network's loss. In GoogLeNet, auxiliary classifiers were used to create a deeper network, but in InceptionV3, an auxiliary classifier acts as a regularizer.

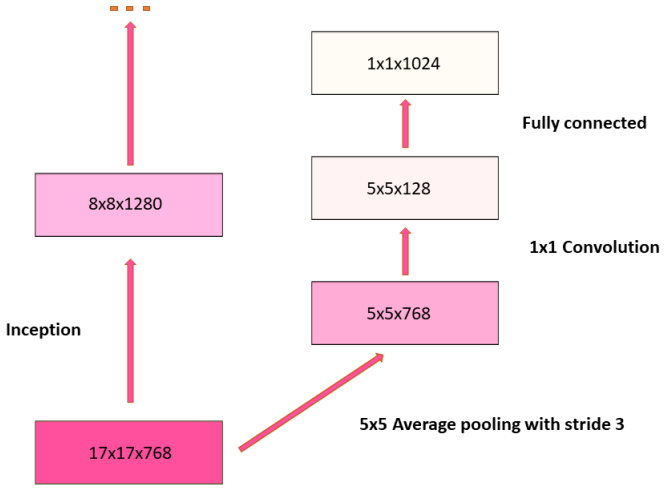

Figure 3.13: Auxiliary classifier.

- **Grid size reduction:** Pooling methods are often used to minimize the size of the grid. However, a more effective technique for circumventing computational cost barriers is presented:

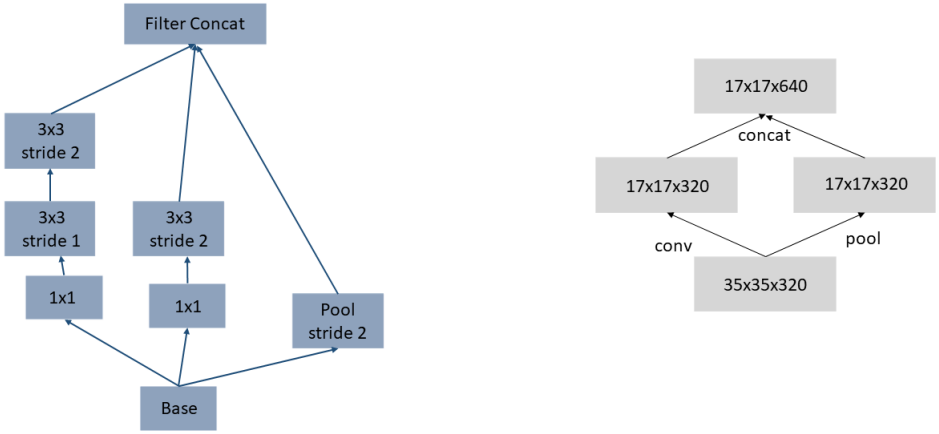

Figure 3.14: Grid size reduction.

- All the above concepts are consolidated into the final architecture.

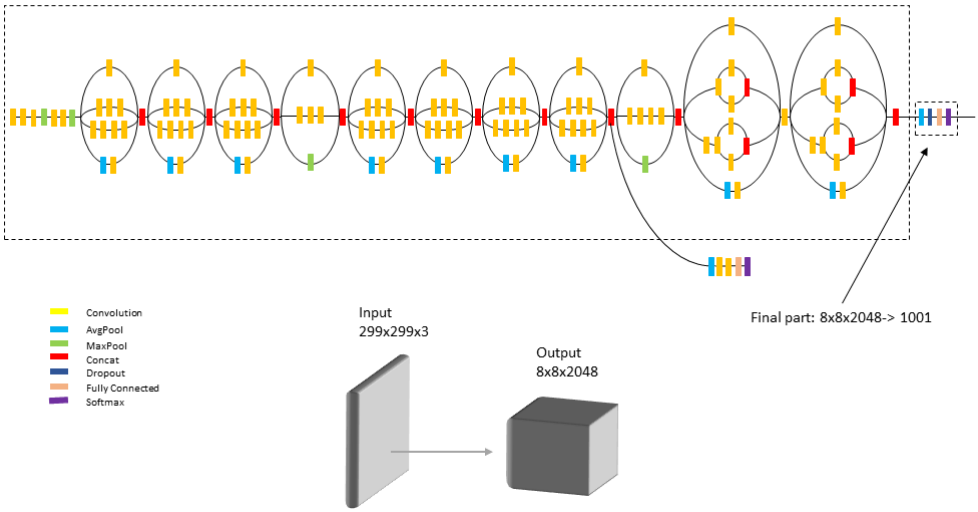

Figure 3.15: Complete architecture of InceptionV3.

The InceptionV3 model is simply an improved version of the InceptionV1 model. The model uses several approaches to optimize the network for more excellent model adaption. It has a more extensive network than the InceptionV1 and V2 models, but its speed is unaffected. Moreover, it is less expensive in terms of computing.
It provides high-performance gain for convolutional neural networks. Effective utilization of computing resources combined with a minor increase in computation load for an Inception network to provide high-performance output is necessary. It can extract characteristics from input data at various scales by using different convolutional filter sizes. Additionally, within the network, 1×1 convolutions lower the dimension of inputs. Because 1×1 convolutions are set up with fewer filters, the outputs often have fewer channels than the initial input. 1×1 Convolution filters learn cross-channel patterns, enhancing the network's overall feature extraction capabilities.
InceptionV3 primarily focuses on burning less processing power by changing earlier Inception architectures. On the ImageNet dataset, InceptionV3 is often used as it has successfully gained an accuracy of 78.1 percent. In 1×1 convolutions, it reduces the dimensions of data traveling through the network, which has the added benefit of expanding the network's width and depth wherein 3×3 and 5×5 convolutions because of the multiple Convolution filter sizes; the network may learn various spatial patterns at different scales.

3.2 Feature Importance Visualization

The term "feature importance" refers to strategies that value input features entirely on their predictive power for a target variable. Feature importance scores are critical components of predictive modeling projects because they give insight into the model and data and serve as the foundation for reducing dimension and choosing the feature, which will improve the use of the predictive model on any problem.

3.2.1 LIME

Deep neural networks are incredibly complicated, and their judgments might be challenging to comprehend. The LIME method uses a smaller, more interpretable model, such as a regression tree, to imitate the classification behavior of a deep neural network. The neural network's decisions may be deduced by interpreting the simpler model's decisions. The basic model is used to determine the relevance of input data features as a proxy for the deep neural network's importance of the features.
When a specific feature is critical to a deep network's classification decision, eliminating that feature significantly impacts the classification score. As a result, characterization is also crucial in the basic model. The imageLIME function in Deep Learning Toolbox is used to construct maps of feature significance defined by the LIME approach. The LIME algorithm for images works by:

- Segmenting an image into features.
- Creating a significant number of synthetic images by randomly inserting or omitting attributes. Excluded features have every pixel replaced with the image average value, thus they no longer carry network-relevant information.
- Classifying the synthetic images with the deep network.
- Fitting a simplified regression model employing the presence or absence of image characteristics as binary regression predictors for target class scores for each synthetic image. In the region of the observation, the model approximates the behavior of the sophisticated deep neural network.
- Using the simple model, computing the significance of features and translating this feature importance into a map that identifies the sections of the picture that are most relevant to the model.

It works by splitting the picture into superpixels and then providing details. The Pixels were divided into groups with similar information and information about any specific portion of a photograph. After this, unwanted images were grouped by not sharing some random superpixels.
Following that, the impact of picture disturbances on the probability of correctly predicting a class was determined. Following that, the adjusted data were used to train a linear model. Similar superpixels for a specific classification were provided as weight values: positive data specifies an effect on classification accuracy, while negative values indicated the opposite.
The following diagram depicts the LIME method's four stages. To Explain this type of sorting, the initial provided picture is split into superpixels. The splitting of updated pictures was created, and classification probabilities were computed using the original prediction model. These probabilities and changed images were entered into a regression model, which determined the contribution of each superpixel to classification as positive or negative. The regression weights are often represented visually through a blue-red color map.

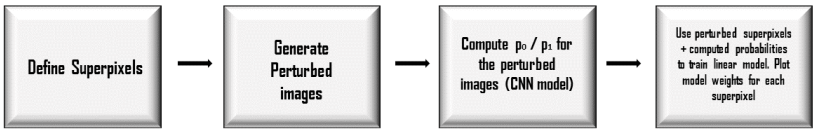

Figure 3.16: LIME algorithm in four steps.

The linear regression weights are often represented visually using a blue-red color map, with blue/red pixels denoting positive/negative consequences. The absolute value of the importance is proportionate to the color intensity. Thus, blue areas indicate superpixels that aid in proper classification, while red areas demonstrate the other. Thus this execution of LIME helps the user to give rise to explanations for isolated classes.
The separation methods used to create superpixels considerably influence the subsequent clarification of the suggested XAI approach. LIME's most recent release contains three segmentation algorithms drawn from the scikit-image Python package.
These algorithms are as follows:

- Felzenszwalb's [1] efficient graph-based image segmentation (FHA) over-segments an RGB picture using tree-based clustering. The initial paramenter is somehow used to define the number of superpixels generate is "scale," thus gives us the monitoring level.

- Achanta et al. [3] segment the picture utilize Simple Linear Iterative Clustering by utilizing K-means congregate in the color space (SLIC). A parameter called "number of segments" that attempts to approximate the quantity of superpixels produced.

- Quickshift, developed by Vedaldi et al. [2], created Quickshift, which does segmentation by grouping pixels using a quick-shift mode seeking method. There is no single parameter that can be utilized to control the number of superpixels in the final image.

Additionally, each of the three approaches includes a "sigma" option for specifying the width of a Gaussian preprocessing phase. Greater sigma values often result in fewer segments, which identifies standard limitation split by all algorithms. Because the usual sigma value for FHA in the library is 0.8, this value was also used in most experiments using the other two techniques.
To provide a more structured contrast of the clarification, this experiment used the pictures with the same number of superpixels generated by all techniques. To do

this, segmentation parameters must be fine-tuned. Because quick-shift allowed for the least flexibility in terms of segment length, utilized as a foundation, to produce the same expected superpixels but along with other two perfect algorithms.
Algorithms' sensitivity to texture and color change and their related factors, the problem arises of how the optimal group of parameters will be constructed. While this is somewhat arbitrarily chosen, the plan is the definitive collection of framework that indicates the most relatable explanations. Using conventional segmentation parameters alone may not result in a meaningful description. As an alternative option, a method for segmenting every picture into grids of 9, 16, 36, 64, 144, 256, and 576 equally-sized squares was devised. While individual segments lacked the context-sensitive relevance of a conventional superpixel, they guaranteed that every image was split into equal parts in the same destinations.
These square segments were then used in the same manner as conventional superpixels in the LIME algorithm. The weight heat maps developed indicated which picture subregions were more relevant for a specific categorization. Perfect grids with a more significant amount of little square segments provided a more detailed view.
Combining the heat maps from all grids created a final thermal map. Although this approach is unlikely to offer as extensive an explanation of too complicated superpixel algorithms, the approximate foundation may give a reasonable rough independent of parameter modification. It is also more straightforward to grasp the pictures due to the guaranteed number of segments. For brevity, this strategy will be referred to as "square grid" throughout this text.
LIME is model agnostic, allowing it to be used with any machine learning model. It approaches the model as a black box, which means that the only way to learn how it works is to alter the input and observe the predictions. LIME modifies the values of features in a single data sample and then assesses the effect on the output. LIME produces a list of explanations for each characteristic's contribution to the data sample prediction. This permits local interpretation and identifies the features that will have the most significant influence on the projection.

Figure 3.17: Concept of the Black Box.

Lime assumes a black box machine learning model and investigate the relationship between the input and output, represented by the model. Users must first and foremost be able to comprehend the explanations, which is not always the case for

the attribute place accepted by the model, as it will contain unlimited input data (even a simpler model with hundreds or thousands of coefficients can be difficult to interpret) or may contain too complex/artificial variables. Consequently, the explanations provided by LIME employ a different data representation (interpretable representation) than the original feature space.
LIME creates a local explanation by approximating the black-box model in the instance region to be explained with an interpretable model (for example, a straight model with a few Numerical coefficients). In summary, LIME generates an interpretable prediction explanation from the components of an interpretable model (for example, the coefficients in linear regression) that replicates the black-box model near the point of interest and is trained over a new data representation ensure interpretability. LIME is one of the few systems capable of concurrently processing tabular data, text, and pictures. The approach has been extensively used in text and picture analysis because of the interpretable data structure.

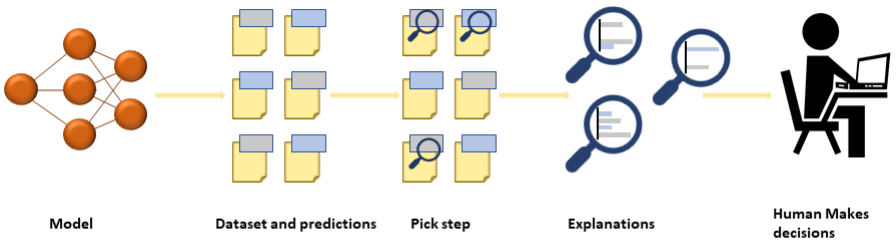

Figure 3.18: Understanding model predictions with LIME.

Lime explains of model predictions at the data sample level. It allows end-users to interpret these predictions and take actions based on them. In this scenario, the explanations are supplied in image/text fragments, and users may quickly find the basis for such descriptions. The method's fundamental concept is simple: a simpler model approximates a more complicated one. Predictions are easier to explain when employing a simpler model with fewer interpretable explanatory variables. Complex, high-dimensional models can be solved using the LIME approach.
The fidelity measure provides us with suitable tools to explain the predictions of the black box with the help of the interpretable model. Models must be explainable to users for humans to trust AI systems. AI interpretability reveals what's going on inside these systems and aids in the detection of potential problems, including information leakage, model bias, robustness, and causality. LIME provides a generic framework for uncovering black boxes and explaining why AI-generated predictions or recommendations are made.

Chapter 4

Dataset Analysis

For our insect pest identification, the IP102 dataset is picked. This dataset has several sub-classes and super-classes. Datasets are divided into two main categories: Field (FC) and Economic Crops (EC). Field Crops are Rice, Beet, Alfalfa, Wheat, and Corn. The Economic Crops have more insects than Field crops usually. The Economic Crops are mainly Mango, Vitis, and Citrus. These are then broken down into more than a hundred child classes, each defining the insect problem linked with the crop. The article also generated some exclusive results for machine learning-based identification. They passed both shallow and deep features using Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) classifiers (SIFT, Gabor, CH, and so on).

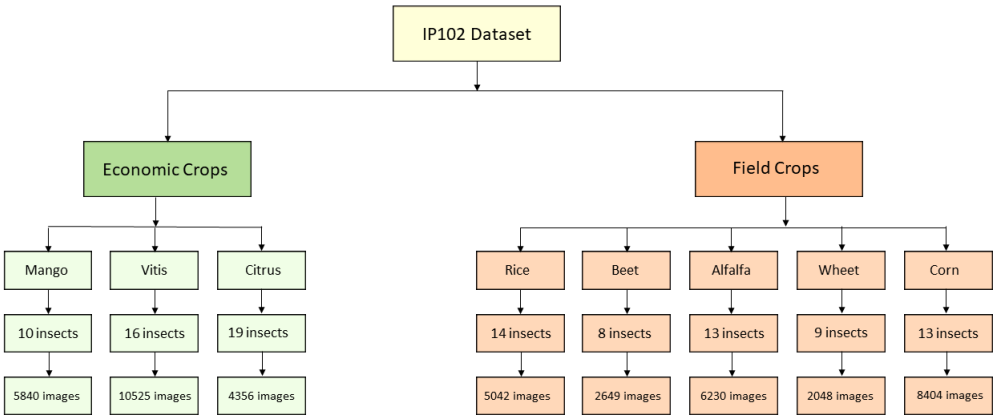

Figure 4.1: IP102 dataset class structure.

The most excellent precision of approximately 70% implies that this research can provide much value. The challenge in the IP102 data set differentiates the rear and front of images are various classes and difficulties. The data set additionally contains an asymmetric sample number for each category. Because of the imbalanced nature, any type with a higher sample size is essential during supervised training, and the classification deviates from the associated class.

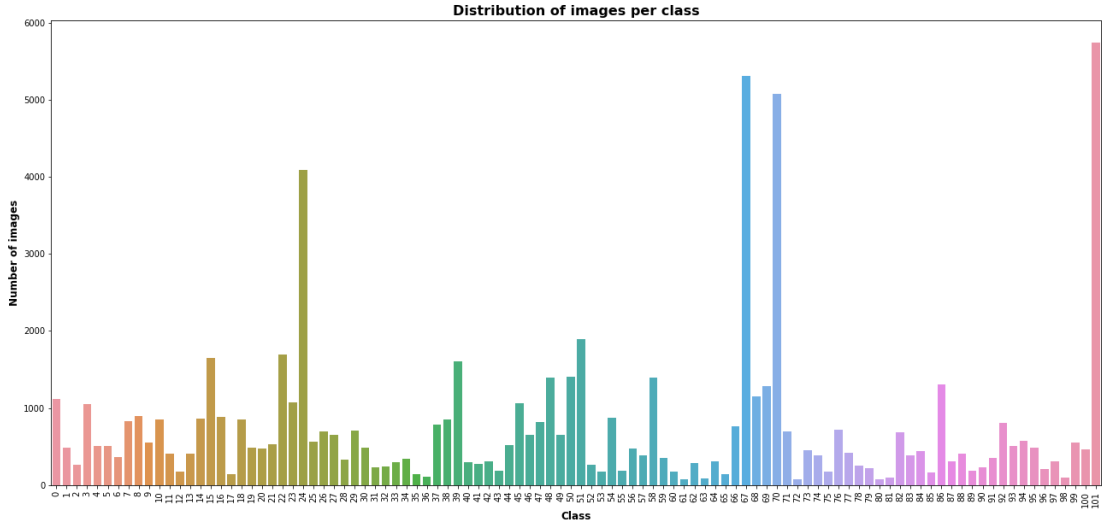

Figure 4.2: Imbalanced class distribution of the IP102 dataset.

The IP102 dataset's training/validation/testing (Train/Val/Test) set split and imbalance ratio (IR) on various class levels. The 'Class' indicates the associated supersub-class class's number. The abbreviations 'FC' and 'EC' stand for field and economic crops, respectively.

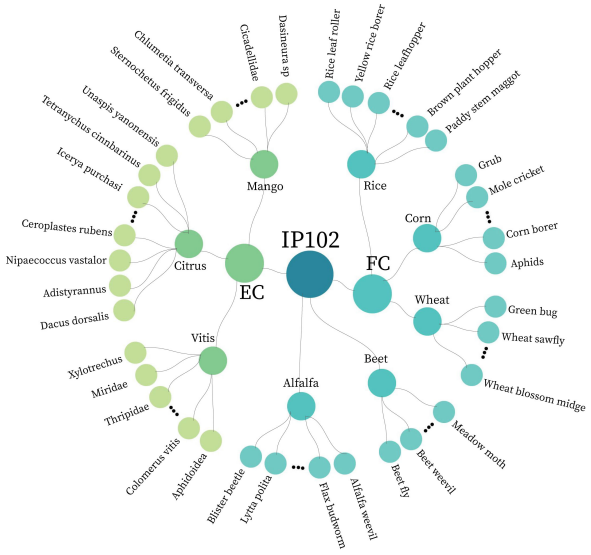

Figure 4.3: Taxonomy of the IP102 dataset.

The abbreviations "FC" and "EC" stand for field and economic crops, respectively. Only 35 classes are presented at the sub-class level.

Table IV

Class Distribution

	Super-Class	Class	Train	Val	Test	IR
FC	Rice	14	5,043	843	2,531	6.4
	Corn	13	8,404	1,399	4,212	27.9
	Wheat	9	2,048	340	1,030	5.2
	Beet	8	2,649	441	1,330	15.4
	Alfalfa	13	6,230	1,037	3,123	10.7
EC	Vitis	16	10,525	1,752	5,274	74.8
	Citrus	19	4,356	725	2,192	17.6
	Mango	10	5,840	971	2,927	61.7
IP102	FC	57	24,602	4,098	12,341	39.4
	EC	45	20,721	3,448	10,393	80.8
	IP102	102	45,095	7,508	22,619	80.8

Table 4.1: Imbalanced class distribution of the IP102 dataset

Table 4.1 summarizes the IP102 dataset's class distribution. First and foremost, the dataset is separated into two sections: FC (Field Crop) and EC (Economic Crop). Second, these two categories are subdivided into various Super-Classes. The FC division has five Super-classes, whereas the EC level has three. When we go further into the table, we can see that each of these Super-Classes has a set number of classes allocated to it. The IP102 dataset's training/validation/testing (Train/Val/Test) set split and imbalance ratio (IR) for various class levels below.
Following the Train, Val, Test, and IR values, the table displays each division's total number of classes. The IP102 dataset has 102 classes in total.

4.1 Data Augmentation

The Deep Neural Network (DNN) models require a great deal of data to avoid oversize because the data set contains an immensely vast number of pictures. However, the data set remains very difficult, particularly for insect classes with small sample photographs. Data is increased by altering the picture vector positions to enhance the information. The risk of skewed data sets lowers, creating more variances for learning DNN models. In recent years, the use of this strategy has been very successful. The results of the data increase are presented below during DNN training.

Table V

Data Augmentation

Augmentation Type	Value Range	Direction
Rotation (In degree)	-30 to 30	Clockwise/ Anti-clockwise
Width shift (In fraction of total width)	-0.05 to 0.05	Left/Right
Height shift (In fraction of total height)	-0.05 to 0.05	Top/Bottom
X axis zoom (In percentage)	0 to 40	X Axis
Y axis zoom (In percentage)	0 to 40	Y Axis

Table 4.2: Data augmentation parameters

Table 4.2 provides a high-level overview of the Data Augmentation Parameters that will be employed. The table contains five different Augmentation Types. Each type of augmentation shows a Value Range and Direction of the particular augmentation type.
First, we can see the Rotation(In degree) augmentation, which has a value range of -30 to 30 and maybe Clockwise or Anti-Clockwise. Second, the Width shift (in a fraction of total width) type, with a value range of -0.05 to 0.05 and a left/right direction. Then there's the Height shift (Infraction of full height). Although it has the same Value Range as the Width shift, its Direction is distinct, being Top/Bottom rather than Left/Right. Following that, we notice the X-axis zoom(In percentage) augmentation type, which has a Value Range of 0 to 40 and a /direction of X-Axis. Finally, the Y-axis is zoomed in (In percentage). Again, it has the same Value Range as the X-axis zoom, but its direction is Y-Axis. Thus, from this table, we get a brief idea of Data Augmentation Parameters.
Images were inverted horizontally at random in addition to the augmentations. Furthermore, all pixel values were scaled from 0 to 1 and labeled as x. The procedure for min-max scaling is as follows:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Furthermore, the validation dataset was the minor component, and it was utilized to evaluate the model after each epoch and adjust the model parameters as needed. However, since parameter optimization was focused only on the validation set, the model may become skewed, favoring the validation set. Consequently, we maintained the separate test set that the model never saw throughout the whole training phase. The model tested on the test set only after the entire training procedure had been completed. After then, the training data was supplemented using the settings listed in table IV to prevent early overfitting. Lastly, all three stages of images had their pixel values scaled.

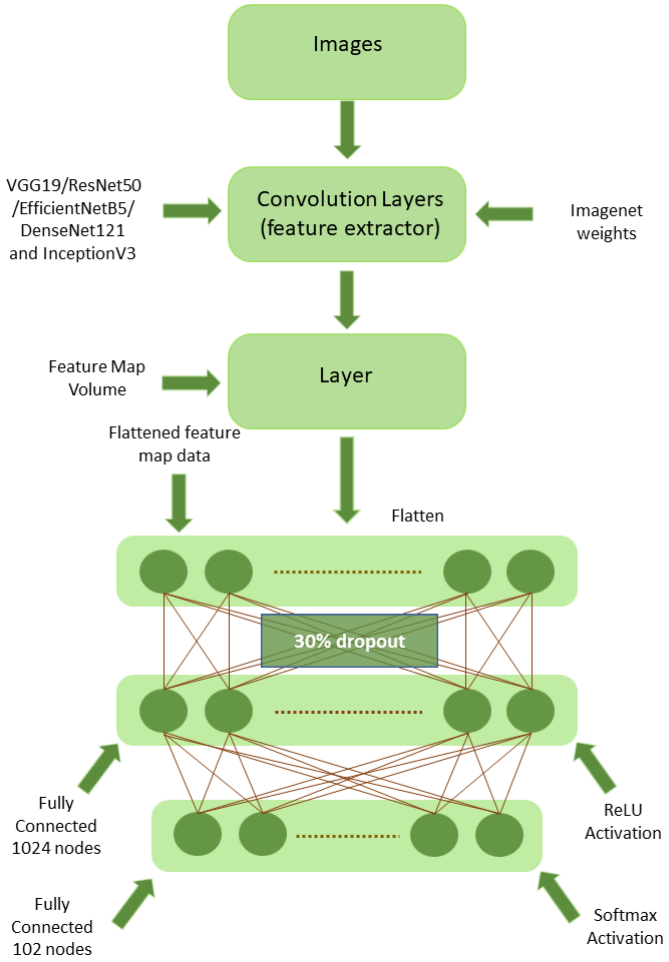

Figure 4.4: Proposed DNN architecture.

The augmented dataset was utilized for training all of the chosen models once it was supplemented and scaled. The models' feature extractor sections were left alone, but the classification layers at the end were removed. Instead, we created two Fully Connected (FC) layers of our own. Each node in the output layer corresponds to a particular class's output. We utilized the weights collected from training models on the ImageNet dataset rather than from scratch with random weights. The gradient descent method was aided in achieving quicker convergence thanks to this setup. Additionally, a 30% dropout was applied to the weights between the flattened feature map and the first FC layer with 1024 nodes to avoid overfitting.

Chapter 5

Implementation and Result Analysis

The implementation of the proposed model for insect pests is described in this section. We ran 30 epochs in the training and validation data sets to calculate accuracy and loss. While training on the complete data set. Before training, all of the pictures were resized to 224×224 pixels. Finally, the calculation was performed on a workstation with an AMD Ryzen 5 3600 3.6 GHz CPU, 16 gigabyte RAM, and RTX 2060 GPU. Despite the continuing improvement in training accuracy, validation accuracy fluctuated around a set figure.

5.1 Performance Metrics

The result of the proposed model for insect pests is described in this section. During training, checkpoints with the highest validation precision were generated, and following the conclusion of the training, the models with the highest validation accuracy were preserved. These models used the entirely unseen test data to be classified.
Calculation of the validation and the test accuracy was done using the following formula:

$$Accuracy = \frac{TP + TN}{NS} \tag{5.1}$$

5.2 Classify into 102 classes

The graphs below show that the training precision starts lower than the validation precision but steadily improves. This can be identified by the vast quantity of training data supplementation and including 30% dropout in an FC convolution layer, making the training part challenging. However, as the training progressed, the length of movement and the correctness of evaluation rose significantly. In the same pattern, the loss of movement and validation happened in the other direction.

Table VI

Validation and Test Accuracy

Model Name	Depth	Parameters	Validation Accuracy	Test Accuracy
VGG19	26	143,667,240	65.58%	69.14%
ResNet50	-	25,636,712	68.84%	70.73%
EfficientNetB5	-	30,562,527	67.59%	70.26%
DenseNet121	121	8,062,504	**70.37%**	**71.98%**
InceptionV3	159	23,851,784	69.12%	70.98%

Table 5.1: Accuracy of the implemented models

Table 5.1 depicts the Validation and Test Accuracy for each model used in our work. For our dataset, we used five alternative models: VGG19, ReNet50, EfficientNetB5, DenseNet121, and InceptionV3. The table also shows the results of Max Validation Accuracy and Corresponding Test Accuracy for each model. From above, we can see that the validation and test accuracy of the following models are all quite closer to each other. However, among the five models, DenseNet121 has the best Max Validation Accuracy (70.37 percent) and Corresponding Test Accuracy (71.98 percent), while VGG19 has the lowest (65.58 percent) and Corresponding Test Accuracy (69.14 percent).
Despite the best and lowest validation and test accuracy results, none of them is less than 65 percent in the case of any deployed models.

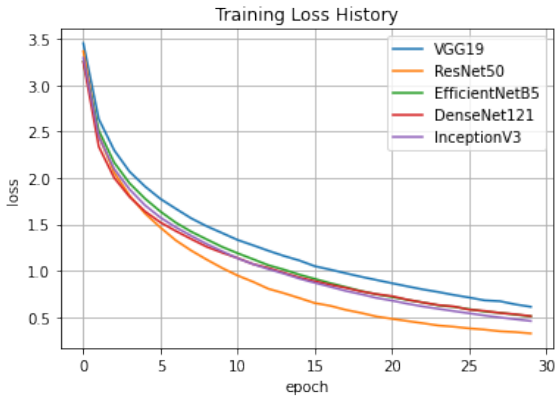

Figure 5.1: Training loss history

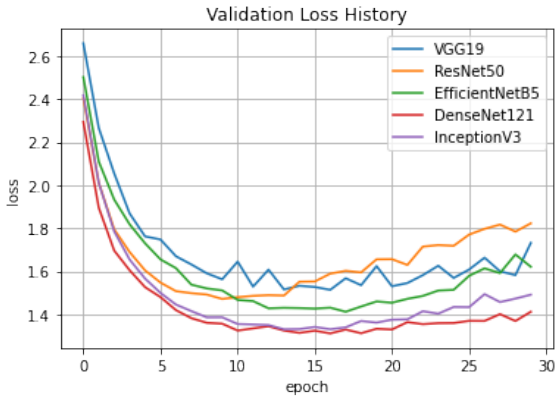

Figure 5.2: Validation loss history

The graph in fig-5.1 and fig-5.2 shows the history of training loss and validation loss in between a range of 30 of VGG19, ResNet50, EfficientNet85, DenseNet121, and InceptionV3 models. The figures are given as a percentage in loss history. Overall, the models fit with training data quite rightly as it has a smooth downward trajectory overtraining loss history. On the other hand, the validation loss history graph shows how the models match if newly acquired data is provided.

Figure 5.3: Training accuracy history

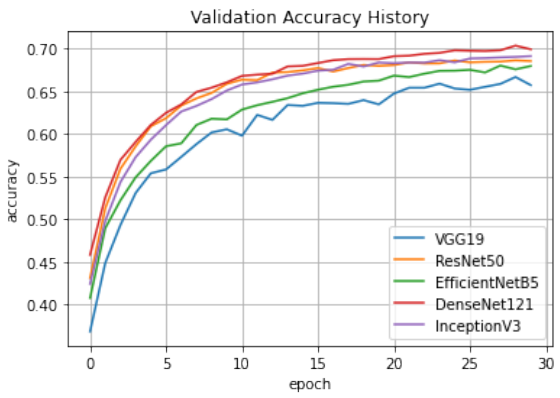

Figure 5.4: Validation accuracy history

Fig-5.3 and 5.4 give an idea about the training and validation dataset. Although the uptake of the training accuracy graph defines that the usage of identical images in both training and testing cases is successful, the uneven validation accuracy graph states the difficulties in the attempt of models to identify and classify the photos correctly.

5.3 Comparing with other research works

We compared our results to the baseline papers [20], Feature Reuse Residual Networks for Insect Pest Recognition [16], High-Performance Ensemble of Convolutional Neural Networks for Insect Pest Image Recognition [36]. Identification of Crop Consuming Insect Pest from Visual Imagery Using Transfer Learning and Data Augmentation on Deep Neural Networks [17] after we trained our models and classified them into 102 classes. The findings of the detailed comparison will be discussed in the next section.

5.3.1 Comparison with the baseline paper [20]

The researchers compiled a large-scale dataset for insect pest recognition called IP102, which included approximately 75,000 photos of 102 species. In their [20] dataset, they also tested various cutting-edge recognition techniques.

Table VII

Comparison with baseline

Model Name	Baseline Accuracy[20]	Implemented Model Accuracy
GoogleNet	43.5%	69.12%
ResNet	48.2%	68.84%
VGGNet	49.4%	65.58%

Table 5.2: Comparing with the baseline accuracy

Table 5.2 portrays the comparison between the baseline accuracy and the accuracy of the implemented models according to our paper. Our research used three models: GoogleNet, ResNet, VGGNet, to compare our work with the baseline. The above table shows that all the implemented models in our profession have more accuracy than the baseline. Among them, GoogleNet has the highest accuracy rate, which increased from 43.5% to 69.12%. However, VGGNet having the lowest increasing accuracy rate, has risen to 65.58% from 49.4% baseline accuracy. As a result, our approach may provide significantly better accuracy than the baseline.

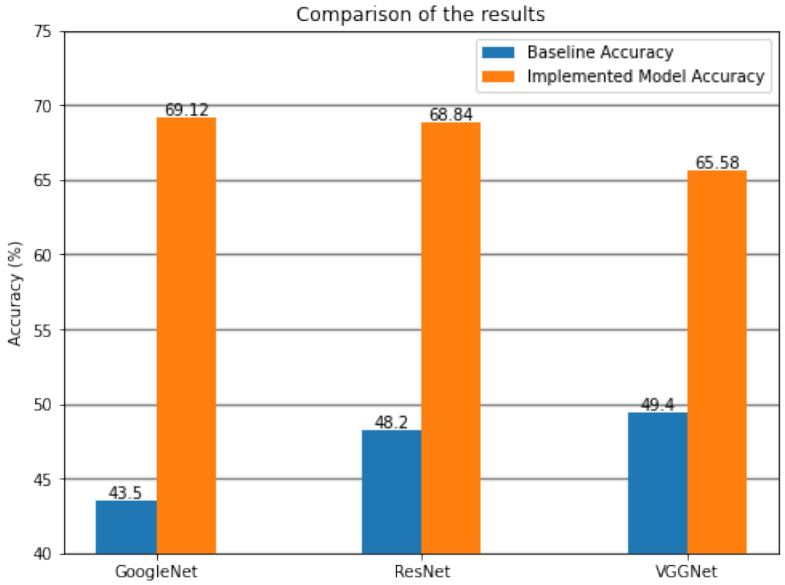

Figure 5.5: Comparing with the baseline accuracy

The bar chart illustrates a comparison of baseline accuracy and implemented model accuracy between GoogleNet, ResNet, and VGGNet. It can be seen that the baseline accuracy grew steadily in each model. It leads to a rise from 43.5% to 48.2% in GoogleNet and ResNet while VGGNet carries the highest value of 49.4%. On the contrary, implemented model accuracy has decreased from 69.12% to 68.84% in GoogleNet and ResNet and continues to reduce up to 65.58% in VGGNet.
The GoogleNet and VGGNet versions used in the study [20] were InceptionV1 and VGG16, respectively, which is an essential factor to note in this regard. Meanwhile, we employed enhanced InceptionV3 and VGG19 versions in our planned research. However, the two study projects used the same ResNet version: the deep ResNet.There are 50 layers. Our study's higher accuracy might be attributed to the more solid architecture provided by these newer models. The improved accuracy with ResNet, in particular, illustrates the importance of feature extraction and dropout enabled classification layers.

5.3.2 Comparison with Paper-A: Feature Reuse Residual Networks for Insect Pest Recognition [16]

Their proposal provided the feature reuse residual network for insect pest detection. [16] It includes learning half of a feature and reusing half of it inside each Residual Feature Reuse block. They built the FR-ResNet and tested its classification ability on the IP102 dataset.
The IP102 dataset utilized the implementation described in the base paper [20]. SGD was used in a 64-piece mini-batch. According to the researchers, the models were based on the training set, and their performance assessed the test set. Pytorch 1.0 and a single Nvidia Titan X were used in their implementations. Researchers created FR-ResNet with varied depths and compared accuracy performance on IP102 to ResNet baseline models in their study. [16] They compared FR-ResNet to several cutting-edge models, including AlexNet, ResNet-50, ResNet101, Googlenet, VGG16, and DeseNet121, using the IP102 dataset. Furthermore, in this proposal, the 34-layer FRResNet outperformed all other models on the test set, with a test accuracy of 54.73 percent. In terms of performance, 50-layer FRResNet outperforms 34-layer FR-ResNet. ResNet-101 has a smaller training loss than ResNet-50, but its test accuracy is inferior owing to overfitting induced by the larger parameters.

Table VIII

Comparison with Paper-A

Model Name	Paper-A Accuracy[16]	Implemented Model Accuracy
GoogleNet	52.17%	69.12%
ResNet	54.19%	68.84%
VGGNet	51.84%	65.58%
DenseNet	54.59%	70.37%

Table 5.3: Comparing with the Paper-A accuracy

Table 5.3 compares the implemented model accuracy with the Paper-A accuracy. Our innovation improved the accuracy of all four models: GoogleNet, ResNet, VGGNet, and DenseNet, as seen in the table. In our model, GoogleNet accuracy has increased to 69.12%, up from 52.17% in Paper-A, whereas VGGNet has a minor improvement, rising from 51.84% to 65.58%. Nevertheless, DenseNet has the overall highest accuracy rate in our paper. Therefore, our implemented models have also more accuracy than Paper-A.

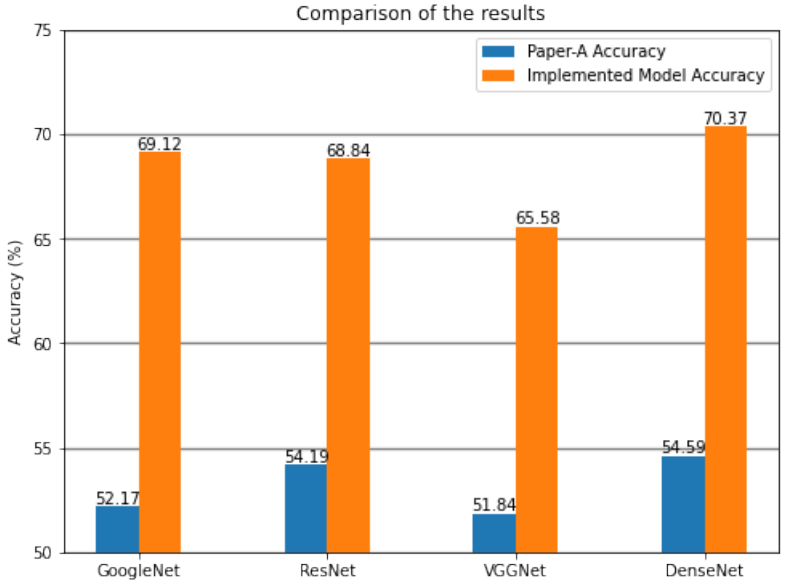

Figure 5.6: Comparing with the Paper-A accuracy

The chart shows the changes in GoogleNet, ResNet, VGGNet, and DenseNet regarding Paper-A's accuracy and implemented model accuracy. In both of them, the lowest accuracy was found in VGGNet, while DenseNet portrays the highest accuracy. It can be seen that ResNet and DenseNet provide higher accuracy 54.19% and 54.59% respectively, in Paper-A, although GoogleNet and DenseNet show better accuracy up to 70.37% (DenseNet) on the implemented model.
They employed two distinct ResNet models in their submission and compared the results, whereas we used only the ResNet-50 in your proposal. The GoogleNet version used in the study [14] was InceptionV1, which is an essential factor to note in this regard. In the interim, we have employed an enhanced InceptionV3 and VGG19 versions in our planned study. The two research programs, however, used the same DeseNet version. As a result, the higher precision of our investigation might be attributed to the more robust architecture provided by these newer models. The improved accuracy with ResNet-50 and DeneNet-121 highlights the importance of feature extraction and dropout enabled classification layers.

5.3.3 Comparison with Paper-B: High performing ensemble of convolutional neural networks for insect pest image detection paper [36]

Researchers [36] integrate CNNs with multiple Adam optimization techniques for pest identification. ResNet50, GoogleNet, ShuffleNet, MobileNetv2, and DenseNet201 are the networks they employ. Two novel Adam algorithms for deep network optimization are proposed based on the Adam variant DGrad. All CNNs for the IP102 dataset were trained with cross-entropy as the loss function and the following parameters: 40 batch size .

Table IX

Comparison with Paper-B

Model Name	Paper-B Accuracy[36]	Implemented Model Accuracy
GoogleNet	64.11%	69.12%
ResNet	65.40%	68.84%
DenseNet	69.74%	70.37%

Table 5.4: Comparing with the Paper-B accuracy

According to our paper, the comparison between the Paper-B accuracy and the accuracy of the implemented models is shown in Table 5.4. To compare our work to Paper-B, we used three models: GoogleNet, ResNet, and DenseNet. The table above demonstrates that all of the models we used in our research were more accurate than Paper-B. GoogleNet has the highest increased accuracy rate, growing from 64.11% to 69.12%, while DenseNet has the lowest accuracy rate, moving from 69.74% Paper-B to 70.37%. Hence, the implemented models will deliver much greater accuracy than paper-B.

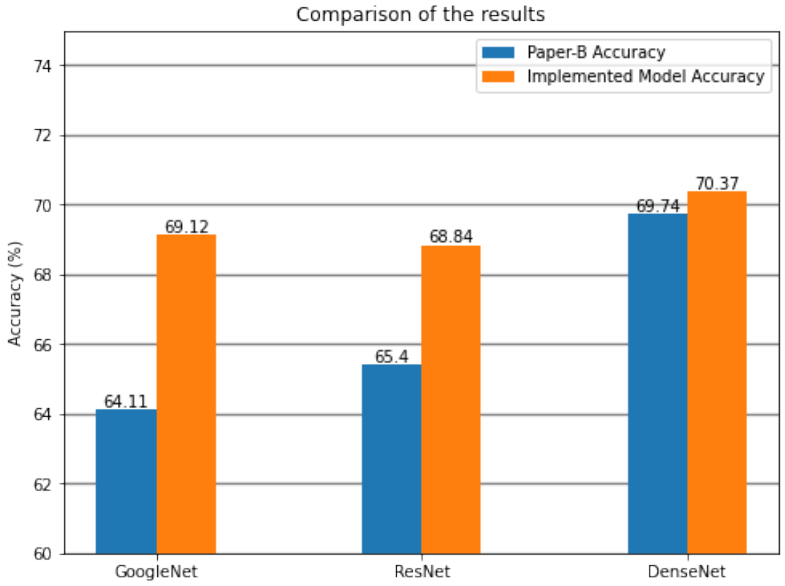

Figure 5.7: Comparing with the Paper-B accuracy

The accuracy rate between the suggested model and research work [36] based on three distinct models are depicted in this graph. DenseNet has the most remarkable accuracy rate, with 69.74 percent [36] and 70.37 percent (for the suggested model), respectively. On the other side, GoogleNet has the lowest accuracy rate for research work [36] 64.11 percent. The most insufficient accuracy was found at 68.84 percent in the ResNet model for our planned work.
InceptionV1 was the GoogleNet version utilized in the study [31], which is essential

to keep in mind. In the meantime, we've been using an improved version of InceptionV3 in our research design. The DenseNet, ResNet version was utilized in both research programs. As a result, the more robust architecture given by these newer models may be responsible for the increased accuracy of our research. The value of feature extraction and dropout enabled classification layers is highlighted by the enhanced accuracy with ResNet-50 and DeneNet-121.

5.3.4 Comparison with Paper-C: Identification of Crop Consuming Insect Pest from Visual Imagery Using Transfer Learning and Data Augmentation on Deep Neural Network paper [17]

The primary purpose of this study [17] was to develop a model for identifying nuisance insects from pictures with the highest level of accuracy. Researchers conducted 30 epochs on both the training and validation datasets in this study to figure out the success. The training and validation accuracy ultimately surpassed 25, but the validation accuracy remained consistent after that. An FC hidden layer may explain this behavior with a substantial training data augmentation and a 50% dropout. With a score of almost 57 percent, Inceptionv3 had the highest accuracy in large-scale categorization. The program also did an excellent job classifying several crop-specific insects, with an accuracy rate of well over 80%.

Table X

Comparison with Paper-C

Model Name	Paper-C Accuracy[17]	Implemented Model Accuracy
GoogleNet	56.73%	69.12%
ResNet	56.35%	68.84%
VGGNet	55.70%	65.58%

Table 5.5: Comparing with the Paper-C accuracy

The accuracy of the proposed model is compared to the accuracy of Paper-C in Table 5.5. As shown in the table, our breakthrough enhanced the accuracy of all three models: GoogleNet, ResNet, and VGGNet. GoogleNet accuracy has improved to 69.12 percent in our model, up from 56.73 percent in Paper-c, whereas VGGNet has improved the least, from 55.70 percent to 65.58 percent. As a result, our implemented models are more precise than Paper-C.

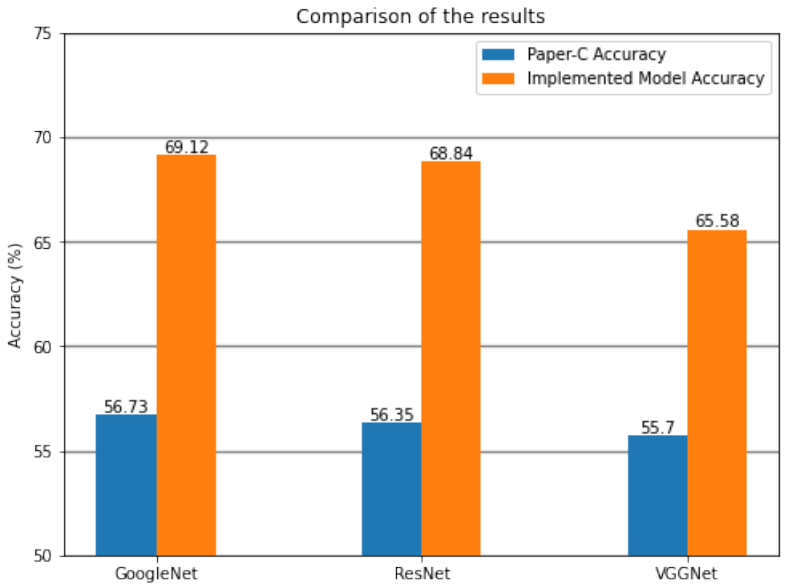

Figure 5.8: Comparing with the Paper-C accuracy

For GoogleNet, ResNet, and VGGNet, the graph displays Paper-C accuracy and implemented model accuracy changes. The accuracy rate of GoogleNet in the implemented model is substantially greater than in paper C, as seen in the graph. In both research work, we can see a considerable difference between the accuracy rate in the case of ResNet and VGGNet models. The accuracy of VGGNet was the lowest, whereas GoogleNet had the highest accuracy.
The discrepancies in accuracy levels are discovered when it comes to data augmentation. The value range for x-axis augmentation in the study article [17] was 0 to 10, and the range for y-axis augmentation was 0 to 25. Our suggested model's value range for x and y-axis augmentation is 0 to 40. In addition to the dropout weights between the flattened feature map and the first layer with 1024 nodes, it assisted us in achieving a higher frequency. To avoid overfitting, the suggested model in the research paper [17] employed a 50% dropout on the weights between the flattened feature map and the first layer FC layer with 1024 nodes. To avoid overfitting, we employed a 30% dropout on the weights between the flattened feature map and the first layer FC layer with 1024 nodes in our suggested models.

5.4 Classify into 8 sub-classes

Finally, although a successful global model is crucial to develop, an insect pest in a real-life setting attracts only one crop. The crop field is, thus, usually sensitive to a relatively limited number, which, according to the crop type, renders the importance of a generic categorization model somewhat irrelevant. Next, we separated the data into eight hierarchical crop categories and categorized all eight components using the best-performed model DenseNet121.

Table XI

Crop based insect classification accuracy

	Super-Class	Max Validation Accuracy	Test Accuracy
FC	Rice	63.06%	74.19%
	Corn	83.49%	81.82%
	Wheat	66.76%	46.31%
	Beet	80.05%	78.79%
	Alfalfa	76.18%	76.99%
EC	Vitis	89.38%	84.03%
	Citrus	83.31%	86.82%
	Mango	92.58%	95.36%

Table 5.6: Crop based insect classification accuracy

Table 5.6 encapsulates crop-based insect classification accuracy, consisting of two separate sections: FC (Field Crop) and EC (Economic Crop). FC is further divided into five super-classes and EC into three super-classes. In the table above, FC has a maximum and minimum test accuracy of 83.49% and 63.06%, respectively, whereas EC has a maximum and minimum test accuracy of 92.58% and 89.38%. Moreover, FC has a maximum and minimum validation accuracy of 92.58% and 83.31%, respectively, whereas EC has a maximum and minimum validation accuracy of 95.36% and 84.03%.

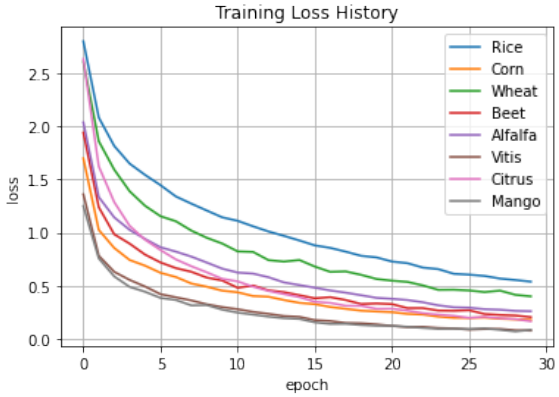

Figure 5.9: Training loss history

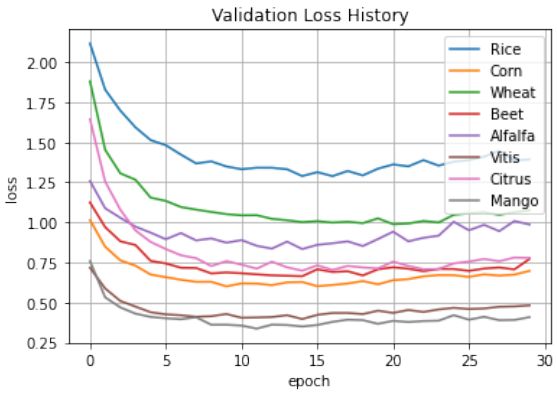

Figure 5.10: Validation loss history

These graphs illustrate the accuracy in insect classification based on various crops like rice, corn, wheat, beet, mango, etc. The percentage of training loss history is relatively evenly distributed among the crops. In contrast, the parallel lines found after the initial stage if newly acquired data is supplied show stability in validation loss history.

Figure 5.11: Training accuracy history

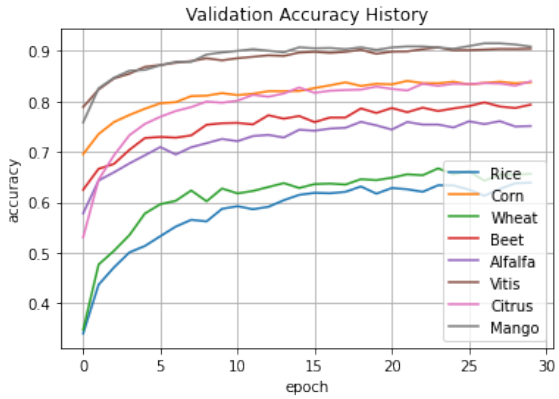

Figure 5.12: Validation accuracy history

In fig-5.11 and fig-5.12 show the difference between the accuracy rate between the training and validation dataset. While the results of training and testing purposes of image identification from different crops have maintained a quite rough upward trajectory, the models can classify the photos based on the produce at a constant rate.

5.5 Comparing with other research work

We compared our findings to the baseline study [20] and Identification of Crop Consuming Insect Pest from Visual Imagery Using Transfer Learning and Data Augmentation on Deep Neural Network [17]. The next section will detail the findings of the in-depth comparison.

5.5.1 Comparison with the baseline paper [20]

We used the same formula shown in (5.1) to calculate the test accuracy of our dataset using DenseNet121.
We ran multiple trials, yet; we ended up with similar results. So, we can say that our models performed better when we separated the data into eight hierarchical crop categories and categorized all eight components using all three models regardless of the model structure, which was identical to the prior parameters.

Table XII

Comparison with baseline

	Super-Class	Baseline Accuracy[20]	Implemented Model Accuracy
FC	Rice	32.1%	63.06%
	Corn	62.2%	83.49%
	Wheat	53.0%	66.76%
	Beet	62.2%	80.05%
	Alfalfa	46.4%	76.18%
EC	Vitis	86.7%	89.38%
	Citrus	76.6%	83.31%
	Mango	89.0%	92.58%

Table 5.7: Comparing with the baseline accuracy

The contrast between the baseline accuracy and the implemented model accuracy of the FC and EC super-classes is shown in Table 5.7. All of the implemented models in the table are more accurate than baseline. Alfalfa accuracy increased the greatest from 46.4% Paper-C accuracy to 76.18% in the applied model, while Corn accuracy grew the least from 62.2% to 83.49%. According to the table, the highest increased accuracy rate among EC is 76.6% in the baseline to 83.31% in the implemented model, while the lowest increased accuracy rate among EC is 89.0% to 92.58% in the implemented model.

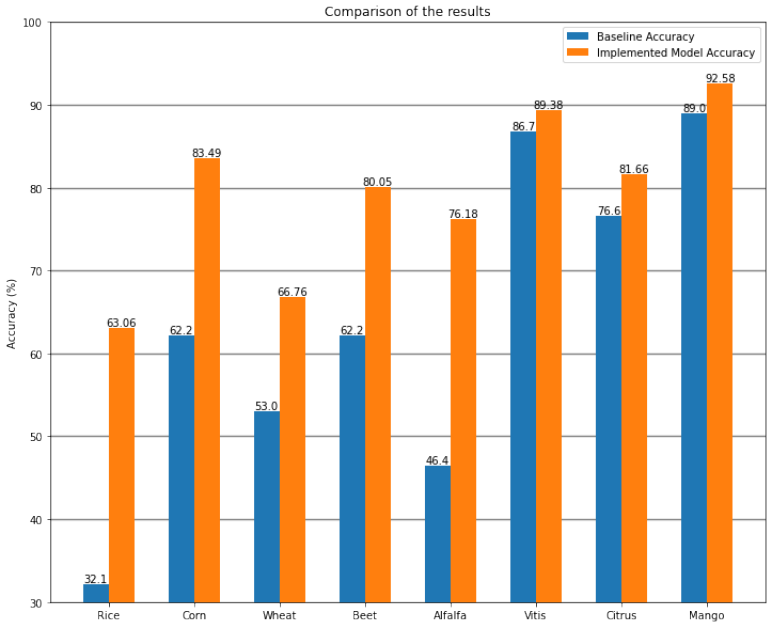

Figure 5.13: Comparing with the baseline accuracy

The graphs compare insect categorization accuracy rates for various crops such as rice, corn, wheat, beet, mango, and others. The accuracy rates were 32.1 percent, 62.2 percent, 53.0 percent, 62.2 percent, 46.4 percent, 86.7 percent, 76.6 percent, and 89.0 percent for the baseline paper. Our suggested model has accuracy rates of 63.06 percent, 83.49 percent, 66.76 percent, 80.5 percent, 76.18 percent, 89.38 percent, 81.66 percent, and 92.58 percent, respectively.
DenseNet121 outperformed the basic model in categorizing economic and food crops. The underlying distinctions between pest classes, the indistinguishability of pests from their environment, and the comparability of pests across various classes contribute to the lack of accuracy. Due to these limitations, it was difficult for the neural network to learn and classify accurately.

5.5.2 Comparison with Paper-C: Identification of Crop Consuming Insect Pest from Visual Imagery Using Transfer Learning and Data Augmentation on Deep Neural Network [17]

They split the dataset based on the eight hierarchical crop classifications in the study publication [17] and used their top-performing model Inceptionv3 to categorize all eight portions. Inceptionv3 was very good at classifying all of the field crops. Insect pest classification findings for crops such as mango, vitis, corn, citrus, and beet were relatively encouraging, while rice, wheat, and alfalfa yielded unsatisfactory results.

Table XIII

Comparison with Paper-C

	Super-Class	Paper-C Accuracy[17]	Implemented Model Accuracy
FC	Rice	52.4%	63.06%
	Corn	79.8%	83.49%
	Wheat	52.7%	66.76%
	Beet	71.0%	80.05%
	Alfalfa	55.8%	76.18%
EC	Vitis	84.0%	89.38%
	Citrus	72.8%	83.31%
	Mango	89.3%	92.58%

Table 5.8: Comparison with the Paper-C accuracy

The contrast between the Paper-C accuracy and the implemented model accuracy of the FC and EC super-classes is shown in Table 5.8. All of the implemented models in the table are more accurate than Paper-C. Alfalfa accuracy increased the greatest from 55.8% Paper-C accuracy to 76.18% in the applied model, while Corn accuracy grew the least from 79.8% to 83.49%. According to the table, the highest increased accuracy rate among EC is 72.8% in the baseline to 83.31% in the implemented model, while the lowest increased accuracy rate among EC is 89.3% to 92.58% in the implemented model.

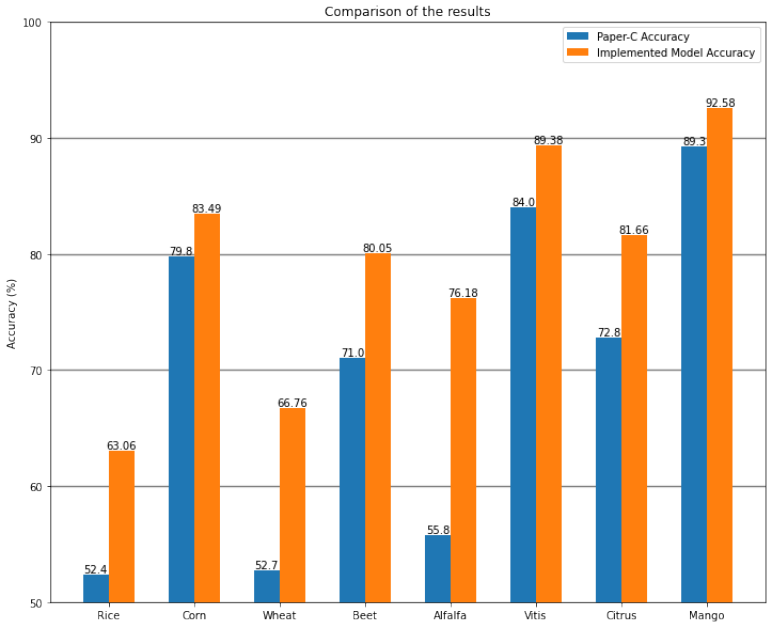

Figure 5.14: Comparison with the Paper-C accuracy

We can see from the graph that the suggested model has greater accuracy. The maximum accuracy was attained by both Paper-C and the proposed model in EC mango (89.3 percent and 92.58 percent, respectively). In comparison, the lowest accuracy was achieved by 'FC' rice (52.4 percent and 63.06 percent, respectively). For both study models, the accuracy rate for the EC subclass is substantially more significant than the 'FC' subclass.
The data was divided into eight hierarchical crop groups by the study model[15] and our recommended model. Where the research paper [17] utilized their best-performing model, Inceptionv3, and we used DenseNet121 in our proposed model. The strong design of DenseNet121 may have contributed to the higher accuracy of our research.

5.6 Understanding model predictions

We attempted to explain the results using explainable AI (Artificial Intelligence). We tried to determine which elements of the photographs are used to classify the insects in the image data. This was accomplished by using LIME (Local Interpretable Model-Agnostic Explanations). We randomly selected five photographs from the 75,000 images to determine why the better-performing model outperformed the others. After randomly selecting five pictures, we ran LIME and obtained the following findings. We end up with the result that is shown in figure 5.15.

Figure 5.15: LIME interpretation results of correct classification

In figure 5.15, the LIME interpretation outcomes are of VGG19 (second row), ResNet50 (third row), EfficientNetB5 (fourth row), DenseNet121 (fifth row), and InceptionV3 (last row). By looking at the results above, we can understand why DenseNet121 outperformed the other models: it seeks to classify insects based on their bodies rather than the entire image, which has an indistinguishable background. It differentiates the insect from its surroundings, even if it is relatively distinct. The remaining models attempted to classify the insects by utilizing the entire image. Although InceptionV3 and ResNet50 got close to DenseNet121 and hence have the second and third highest accuracy rates among the models, we tested. We demonstrated why the best-performing model performed better than the rest by utilizing explainable AI.

However, we also tried to explain why the accuracy did not reach more than 70% even on the highest performing model. For that explanation, we picked five more images randomly from the test dataset and reran LIME on them to discover the reason. We received the following findings. We end up with the outcome that is displayed in picture 5.16.

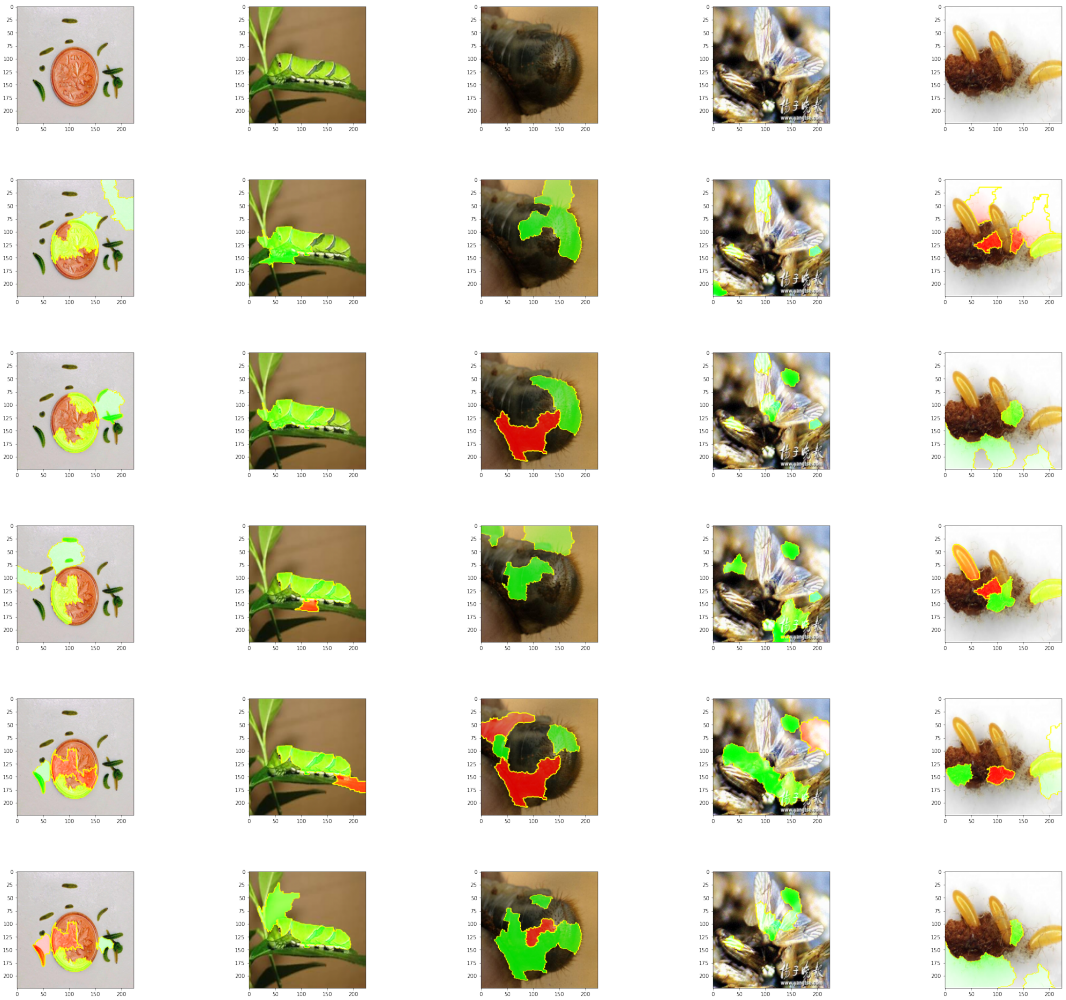

Figure 5.16: LIME interpretation results of mismatch classification

The LIME interpretation results for VGG19 (second row), ResNet50 (third row), EfficientNetB5 (fourth row), DenseNet121 (fifth row), and InceptionV3 (sixth row) are shown in Figure 5.16. By examining the findings above, we can see why the accuracy did not exceed 70% even on DenseNet121. Let's examine the DenseNet121's interpretation closely. We can see that it could not categorize the random five photos, but some of the other models classified them even though they fell short of the 70% accuracy rate. The pictures feature an indistinguishable backdrop from the actual image, and the models have difficulty distinguishing the background from the actual image.

Additionally, there are 102 classifications from which the models became confused about accurately identifying the insects. This constraint may be overcome by increasing data augmentation and training our model on the dataset mentioned above. For these reasons, DenseNet121 was unable to achieve better accuracy rates, and while the others categorized certain insects correctly, they also failed to classify other species, which DenseNet did.

Chapter 6

Conclusion and Future Work

We classified insect pests in this work using a large-scale dataset known as IP102, which contains over 75,000 photos of 102 species. The IP102 complies with several features associated with insect pest distribution in real-world contexts. The suggested research sought to enhance the ability of large-scale crop-eating insect pests to be identified and classified. On the dataset, we investigated many state-of-the-art recognition techniques and presented it using the LIME based XAI framework. However, there is still more work to be done. While enhanced performance was ensured, more enhancements are necessary to provide practical application. With a score of more than 70%, DenseNet121 earned the highest accuracy in large-scale classification. Additionally, the model performed well in classifying several crop-specific insects, with an accuracy of over 80%. This finding suggests that small-scale deployment is currently achievable for various crops. The simplicity of the LIME base framework also serves to improve the usability and comprehension of low-level data. This notion may be applied to various fields of data processing.
Some of the less-than-ideal outcomes are definitely due to a lack of sufficient data, even though we employed the most significant dataset available since there were apparent symptoms of imbalanced datasets during training. If other datasets of these identical insects are developed in the future, they may all be merged to solve the issue. More significantly, the possibility of extensive data augmentation adds an exciting dimension. Given that insects may remain in any orientation or position in their surroundings, the potential for data augmenting makes logical sense, as it enables DNNs to learn from all directions and position patterns. As a result, further data augmentation will be employed to determine whether the models' performance may be improved. Finally, the current version of the research does not address the issue of models producing skewed findings due to imbalanced class distribution. This issue might be solved in the future by including class weights in the training process. We anticipate that our work contributes to the advancement of future research on a variety of essential issues and popular image classification and recognition tasks.

Bibliography

[1] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[2] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *European conference on computer vision*, Springer, 2008, pp. 705–718.

[3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. Venugoban and A. Ramanan, "Image classification of paddy field insect pests using gradient-based features," *International Journal of Machine Learning and Computing*, vol. 4, no. 1, p. 1, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[8] K. Dimililer and S. Zarrouk, "Icspi: Intelligent classification system of pest insects based on image processing and neural arbitration," *Applied Engineering in Agriculture*, vol. 33, no. 4, p. 453, 2017.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[10] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, "Gather-excite: Exploiting feature context in convolutional neural networks," *arXiv preprint arXiv:1810.12348*, 2018.

[11] S.-H. Tsang, "Review: Densenet—dense convolutional network (image classification)," *en línea].[consulta: 7 abril 2019]. Disponible en: https://towards-datascience. com/review-densenetimage-classification-b6631a8ef803*, 2018.

[12] J. Wäldchen and P. Mäder, "Machine learning for image based species identification," *Methods in Ecology and Evolution*, vol. 9, no. 11, pp. 2216–2225, 2018.

[13] Y. Zheng, C. Yang, and A. Merkulov, "Breast cancer screening using convolutional neural network and follow-up digital mammography," in *Computational Imaging III*, International Society for Optics and Photonics, vol. 10669, 2018, p. 1 066 905.

[14] I. P. De Sousa, M. M. B. R. Vellasco, and E. C. Da Silva, "Local interpretable model-agnostic explanations for classification of lymph node metastases," *Sensors (Basel, Switzerland)*, vol. 19, no. 13, 2019.

[15] M. Hassan, "Resnet (34, 50, 101): Residual cnns for image classification tasks," *Neurohive. io*, 2019.

[16] F. Ren, W. Liu, and G. Wu, "Feature reuse residual networks for insect pest recognition," *IEEE Access*, vol. 7, pp. 122 758–122 768, 2019.

[17] M. T. Reza, N. Mehedi, N. A. Tasneem, and M. A. Alam, "Identification of crop consuming insect pest from visual imagery using transfer learning and data augmentation on deep neural network," in *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2019, pp. 1–6.

[18] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.

[19] K. Thenmozhi and U. S. Reddy, "Crop pest classification based on deep convolutional neural network and transfer learning," *Computers and Electronics in Agriculture*, vol. 164, p. 104 906, 2019.

[20] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8787–8796.

[21] V. Agarwal, *Complete architectural details of all efficientnet models*, 2020.

[22] J. G. A. Barbedo, "Detecting and classifying pests in crops using proximal images and machine learning: A review," *AI*, vol. 1, no. 2, pp. 312–328, 2020.

[23] S. A. Burhan, S. Minhas, A. Tariq, and M. N. Hassan, "Comparative study of deep learning algorithms for disease and pest detection in rice crops," in *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, IEEE, 2020, pp. 1–5.

[24] A. Kaushik, "Understanding resnet50 architecture," *OpenGenus Foundation. Retrieved from: https://iq. opengenus. org/resnet50-architecture*, 2020.

[25] V. Kurama, "A review of popular deep learning architectures: Resnet, inceptionv3, and squeezenet," *Consult. August*, vol. 30, 2020.

[26] H. Kuzuhara, H. Takimoto, Y. Sato, and A. Kanagawa, "Insect pest detection and identification method based on deep learning for realizing a pest control system," in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, IEEE, 2020, pp. 709–714.

[27] Y. Liu, L. Wang, J. Cheng, C. Li, and X. Chen, "Multi-focus image fusion: A survey of the state of the art," *Information Fusion*, vol. 64, pp. 71–91, 2020.

[28] A. Mahmood, A. G. Ospina, M. Bennamoun, *et al.*, "Automatic hierarchical classification of kelps using deep residual features," *Sensors*, vol. 20, no. 2, p. 447, 2020.

[29] M. M. S. Maswood, T. Hussain, M. B. Khan, M. T. Islam, and A. G. Alharbi, "Cnn based detection of the severity of diabetic retinopathy from the fundus photography using efficientnet-b5," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, 2020, pp. 0147–0150.

[30] E. K. Meineke, C. Tomasi, S. Yuan, and K. M. Pryer, "Applying machine learning to investigate long-term insect–plant interactions preserved on digitized herbarium specimens," *Applications in Plant Sciences*, vol. 8, no. 6, e11369, 2020.

[31] L. Nanni, G. Maguolo, and F. Pancino, "Insect pest image detection and recognition based on bio-inspired methods," *Ecological Informatics*, vol. 57, p. 101 089, 2020.

[32] J. Peng, S. Kang, Z. Ning, *et al.*, "Residual convolutional neural network for predicting response of transarterial chemoembolization in hepatocellular carcinoma from ct imaging," *European radiology*, vol. 30, no. 1, pp. 413–424, 2020.

[33] Z. Shi, H. Dang, Z. Liu, and X. Zhou, "Detection and identification of stored-grain insects using deep learning: A more effective neural network," *IEEE Access*, vol. 8, pp. 163 703–163 714, 2020.

[34] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep learning for edge computing applications: A state-of-the-art survey," *IEEE Access*, vol. 8, pp. 58 322–58 336, 2020.

[35] T. Kasinathan and S. R. Uyyala, "Machine learning ensemble with image processing for pest identification and classification in field crops," *Neural Computing and Applications*, pp. 1–14, 2021.

[36] L. Nanni, A. Manfe, G. Maguolo, A. Lumini, and S. Brahnam, "High performing ensemble of convolutional neural networks for insect pest image detection," *arXiv preprint arXiv:2108.12539*, 2021.