

चावल और गेहूं के उच्च.श्रूपुट फेनोटाइपिंग के  
लिए चित्र.विश्लेषण एल्गोरिदम्स

IMAGE ANALYSIS  
ALGORITHMS FOR HIGH-  
THROUGHPUT PHENOTYPING  
OF RICE AND WHEAT

TANUJ MISRA  
(10687)



*ICAR-Indian Agricultural Statistics Research Institute*  
*ICAR-Indian Agricultural Research Institute*  
*New Delhi - 110012*  
*2020*

# IMAGE ANALYSIS ALGORITHMS FOR HIGH-THROUGHPUT PHENOTYPING OF RICE AND WHEAT

BY  
TANUJ MISRA

*Thesis submitted to the Faculty of Post-Graduate School,  
Indian Agricultural Research Institute, New Delhi,  
In partial fulfillment of the requirements for the degree of*

*Doctor of Philosophy  
in  
Computer Application*

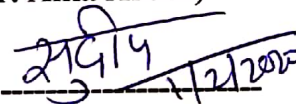
Approved By:

Chairman:

 11/2/2020

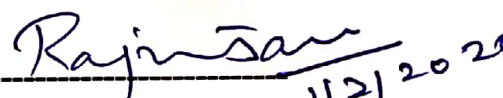
(Dr. Alka Arora)

Co-chairman:

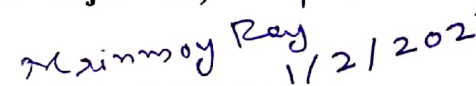
 11/2/2020

(Dr. Sudeep Marwaha)

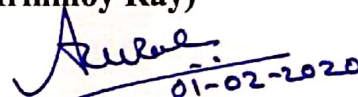
Members:

 11/2/2020

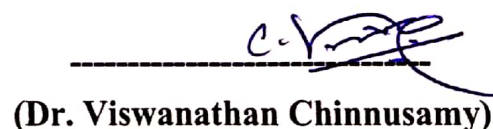
(Dr. Rajni Jain)

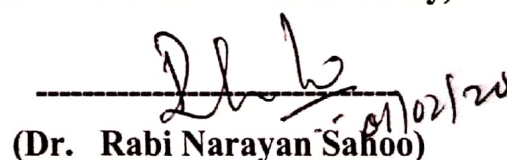
 11/2/2020

(Dr. Mrinmoy Ray)

 01-02-2020

(Dr. A. R. Rao)

  
(Dr. Viswanathan Chinnusamy)

 01/02/20  
(Dr. Rabi Narayan Sahoo)



ICAR-Indian Agricultural  
Research Institute  
New Delhi - 110012



Dr. Alka Arora  
Principal Scientist  
Division of Computer Application  
ICAR-IASRI, Pusa  
New Delhi-110012

### CERTIFICATE

This is to certify that the thesis "Image Analysis Algorithms for High-throughput Phenotyping of Rice and Wheat" submitted in the partial fulfilment of the requirement for the DOCTOR OF PHILOSOPHY in COMPUTER APPLICATION of Post-Graduate School, ICAR-Indian Agricultural Research Institute, New Delhi, is a record of bonafide research work carried out by Mr. Tanuj Misra under my guidance and supervision. No part of the thesis has been submitted for any other degree or diploma.

All assistance and help received during the course of this investigation has been duly acknowledged by him.

Date: 11/21/2020  
New Delhi-12

  
(Dr. Alka Arora)

Chairman,  
Advisory Committee

## **Acknowledgements**

*This thesis came to fruition due to the knowledge gained over the entire period of my Ph. D. study at ICAR-IASRI during which I have been in touch with a great number of people whose contribution in varied yet myriad ways led to the research and making of the thesis which deserve special mention. It is a pleasure to convey my gratitude to all of them by way of my humble acknowledgements.*

*In the first place, I express my deep sense of gratitude to **Dr. Alka Arora**, Principal Scientist, ICAR-IASRI, New Delhi and Chairperson of my Advisory Committee for her initiative, benevolence, endurance, constructive criticism and constant monitoring during the period of my investigation and also in the preparation of this thesis. Above all and the most needed, she provided me the unflinching encouragement and support in various ways. I consider myself fortunate in having the privilege of being guided by her. I am indebted to her more than she knows.*

*I express my deep sense of gratitude to **Dr. Sudeep Marwaha**, Professor and Head (A), Division of Computer Application, ICAR-IASRI and also the co-chairperson of my advisory committee. I am very much grateful for his valuable suggestions and punctilious criticism of my research work. He worked equally hard with me to give this work a presentable shape.*

*I am equally indebted to **Dr. Rajni Jain**, ICAR-National Institute of Agricultural Research and Policy Making (ICAR-NIAP), New Delhi and member of my Advisory Committee for her valuable suggestions.*

*I am extremely thankful to **Dr. A. R. Rao**, Professor, Division of Bioinformatics, ICAR-IASRI, New Delhi and **Dr. Mrinmoy Ray**, ICAR-IASRI, New Delhi, members of my Advisory Committee for their valuable suggestions and constant encouragement during my investigation.*

*I am also highly grateful to **Dr. V. Chinnusamy**, Head, Division of Plant Physiology, ICAR-IARI, New Dehil, and **Dr. R. N. Sahoo**, Principal Scientist, ICAR-IARI, New Delhi, members of my Advisory Committee for their counsel, valuable suggestions and willingness to help for accomplishment of my research work.*

*I am grateful to the help provided by **Dr. Sudhir Kumar**, Scientist, ICAR-IARI, New Delhi and **Dr. Dhandapani R.**, Scientist ICAR-IARI, New Delhi.*

*I duly acknowledge the services of Nanaji Deshmukh Plant Phenomics Center, ICAR-IARI, New Delhi for collection of image data from the experiments. I thank all the members of this center for their timely help and supports.*

*I express my deep sense of gratitude to **Dr. Aditya Nigam**, Assistant Professor, School of Computer Science and Electrical Engineering (SCEE), Indian Institute of Technology, Mandi and one of his Ph.D. Scholar **Ranjeet Jha**. I am very much grateful for their valuable suggestions in the area of Deep Learning and punctilious criticism of my research work.*

*I am deeply indebted to **Dr. Tauqueer Ahmad**, Director (A), ICAR-IASRI for his constant help and encouragement throughout the investigation.*

*I extend my sincere thanks to the Dean and also to the Director, ICAR-IARI, New Delhi and staff of PG School for their helpful attitude and cooperation, throughout the period of study.*

*Fellowship awarded by ICAR-IASRI is duly acknowledged.*

*Collective and individual acknowledgments are owed to my seniors, batch mates and juniors whose presence was somehow perpetually invigorating, helpful and memorable.*

*I take this opportunity to appreciate the help rendered by the staff of TAC, and CAS Lab, ICAR-IASRI. Special thanks are due to Sanjeev Sir, Sunil Sir and Gagan ji.*

*Love, encouragement, and enthusiasm are the fabric stories of intellectual foundation. This is the point, which I have inculcated with the blessing of my family. I bow my mind, body and spirit to be sheltered by the warmest, glorified, golden rays of my beloved parents and all other family members without that affection and moral support it would not have been possible to reach at this stage of my career.*

*Finally, I would like to thank everybody who was imperative to the successful realization of this thesis, as well as articulate my apology that I could not mention personally one by one.*

**Date:** 1/2/2020

**Place:** IASRI, New Delhi-110012

**(Tanuj Misra)**

# CONTENTS

Title	Page No.
<b>Chapter-I Introduction</b>	<b>1-8</b>
1.1 Background	1
1.2 Phenomics and high-throughput plant phenotyping facility	1-2
1.3 Major cereal crops & yield related traits	2-3
1.4 Artificial intelligence & Machine learning	3-4
1.5 Problem definition and Motivation	4-6
1.6 Objectives	6
1.7 Plan of thesis	6-8
<b>Chapter-II Review of Literature</b>	<b>9-17</b>
2.1 Introduction	9
2.2 Plant phenotyping through digital image analysis	9-15
2.3 Application of deep learning techniques in agriculture and allied sectors	15-16
2.4 Plant phenotyping facility	16-17
<b>Chapter-III Materials and Methods</b>	<b>19-47</b>
3.1 Introduction	19
3.2 Image analysis based Leaf Fresh Weight (LFW) estimation in rice plant	19-26
3.2.1 Image acquisition for LFW estimation	19-20
3.2.2 Proposed approach of LFW estimation - VN_LFW	20-24
3.2.3 Model development and performance measurement of the proposed VN_LFW approach	24-25
3.2.4 Tools used in implementing VN_LFW approach of LFW estimation	25-26
3.3 Spike identification and counting in wheat plant through digital image analysis	26-47
3.3.1. Image acquisition for wheat plants	27
3.3.2 Proposed approach of spike identification and counting	27-39
3.3.3 Model development and performance measurement of the proposed approach of spike identification and counting	39-42
3.3.4 Tools used in implementing the proposed methodology of spike identification and counting	42-47
<b>Chapter-IV Results and Discussion</b>	<b>49-78</b>
4.1 Introduction	49

4.2 Analysis of VN_LFW approach of LFW estimation	49-54
4.2.1 Model development and performance evaluation of VN_LFW approach	52-54
4.3 Analysis of the proposed approach of spike identification and counting in wheat plant	55-77
4.3.1 Deep learning based approach for spike identification	55-61
4.3.1.1 SpikeSegNet-- Spike Segmentation Network	55-61
4.3.1.2 LGspikeNet -- Local patch extraction and Global mask refinement spike detection Network	61-71
4.3.2 Performance measure of spike count approach	71-74
4.3.3 Online software for spike identification and counting	74-77
<b>Chapter-V Summary and Conclusion</b>	<b>79-82</b>
<b>Abstract (English)</b>	<b>83</b>
<b>Abstract (Hindi)</b>	<b>85</b>
<b>References</b>	<b>87-94</b>
<b>Appendix</b>	<b>i-xxvi</b>
Appendix I	i-iv
Appendix II	v-xxvi
<b>Annexure</b>	<b>i-ix</b>
Annexure I	i-ix
Annexure II	x-xiv

# CHAPTER I

## INTRODUCTION

---

### 1.1 Background

Under challenging environmental situation, significantly improved crop varieties are needed to cope up with the rapidly growing human population scenarios (Furbank *et al.*, 2009; Sticklen, 2007). It is the greatest challenge to the agricultural scientists and policy makers to meet the future demand of agricultural production under challenging environmental situations (Bruinsma, 2003). For this purpose, systematic quantification of phenotypic traits or components of a particular genotype in a given environment is necessary. Plant phenomics is the study of plant growth, and performance based on morphology, physiology and phenotypic traits or characteristics of the plant. In conventional methods, these traits are recorded either manually or visually which is not only time-consuming and labour-intensive but may also be error prone to acquire large amount of dataset. Therefore, focus has been shifted on precise, accurate and rapid phenotyping for the last few years. In this context, high-throughput image analysis ([www.plantphenomics.org.au](http://www.plantphenomics.org.au); Furbank *et al.*, 2009; Finkel, 2009; Jansen *et al.*, 2009; Klukas *et al.*, 2014; Knecht *et al.*, 2016) is being used to extract several phenotypic parameters related to plant growth, development, tolerance, resistance, architecture, physiology, ecology, yield and the basic measurement of individual quantitative parameters that form the basis for the more complex traits. In this manner, the tedious and time-consuming manual analyses of phenotypic traits are reduced.

### 1.2 Phenomics and high-throughput plant phenotyping facility

The word “*phenome*” refers phenotype (Soule, 1967) which means expression of a genome for certain traits in a given environment. Phenomics is used as an analogy to the genomics that deals with large amount of high-dimensional data. Plant phenomics is the study of plant growth, performance and composition on the basis of morphology, biochemical and physiological traits or characteristics of the plant. Conventional measurement of these traits are recorded either manually or visually which is not only time-consuming and labour-intensive but may also be

error prone to acquire large amount of dataset. Therefore, focus has been shifted on precise, accurate and rapid phenotyping for the last few years.

In this perspective, high-throughput plant phenotyping platforms using non-invasive imaging technologies have emerged as an important area to provide efficient data management, image analysis and result visualization of large-scale plant phenotypic datasets. It is non-invasive as no physical instruments have been directly used to measure plant phenotypic parameters in destructive way. The example of high-throughput plant phenotyping platforms are Integrated Analysis Platform (IAP) developed by Klukas *et al.* (2014), Image Harvest (IH) developed by Knecht *et al.* (2016), LemnaTec (GmbH, Aachen, Germany) *etc.* Non-invasive imaging technologies include visual (or RGB) imaging, near infra-red (NIR) imaging, fluorescence imaging *etc.*

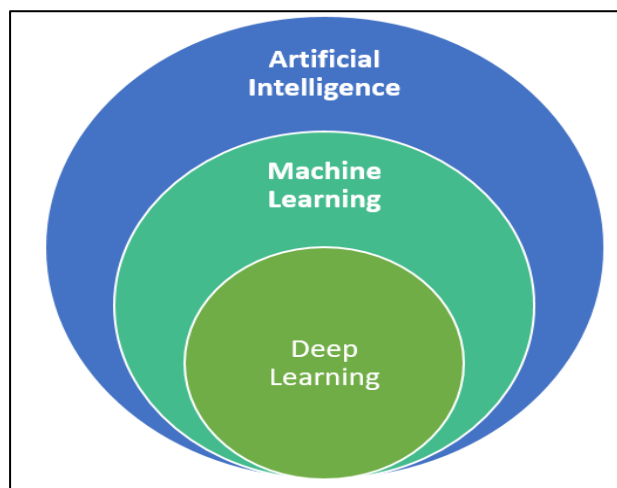
### **1.3 Major cereal crops & yield related traits**

Rice (*Oryza sativa*) and wheat (*Triticum aestivum*) are considered as major cereal crops. Urbanization and rising incomes are driving a rapid rise in the global rice and wheat consumption. Besides, there exist several constraints like drought, biotic and abiotic stresses, climate changes *etc.* which affects the global production of the crops. Therefore, genetic improvement of rice and wheat genotype for input use efficiency and climate resilience is the key for future food security. Plant biomass, leaf area, chlorophyll content, number of spikes or panicle *etc.*, plays a vital role in the study of functional plant biology, growth analysis and are considered as determining factor of net primary production of the crop. Leaf fresh weight (LFW) is used to estimate yield as well as plant biomass (Poorter and Nagel, 2000; Niklas and Enquist, 2002) and are measured by using gravimetric weighing of the harvested sample leaves. Besides, yield is measured in terms of grain which is found within the spikes/panicles in the cereal plants. So, counting of number of spikes can be an important measure to determine yield of the crop. Spike/panicle is an important agronomic component (Jin *et al.*, 2017) which is not only closely associated with yield, but also plays an important role in nutrition examination, and growth period determination. Conventional measurement of the above mentioned yield related traits are destructive, laborious and time-consuming. In this context, several literatures are available on non-destructive measurement of these traits through image analysis

[Qiongyan *et al.* (2017); Sadeghi-Tehran *et al.* (2017); Hasan *et al.* (2018)]. Most of the literature reveals that, current trend in image-based plant phenotyping will be a combined effort of image processing and machine learning technique for feature extractions and data analysis purpose (Tsafaris *et al.*, 2016).

#### 1.4 Artificial intelligence & Machine learning

In computer science, artificial intelligence (AI) or machine intelligence is the intelligence demonstrated by machines in a similar manner as an intelligent human can think. According to John McCarthy, father of AI, “*it is the science and engineering of making intelligent machines, especially intelligent computer programs*”. The main aim of AI is to program the computer for certain traits such as knowledge, reasoning, learning, planning, problem solving *etc.* Machine learning is a subset of AI which uses statistical techniques to enable the machine to perform some specific tasks. It follows “*learning by example*” principle. The methods of learning can be categorized into three types: (a) **supervised** learning algorithm is given with labelled data and the desired output whereas (b) **unsupervised** learning algorithm is given with unlabelled data and identifies the patterns from the input data and (c) **reinforcement** learning algorithm allows machines and software agents to determine the ideal behaviour automatically within a specific context, in order to maximize its performance.



**Fig.1.1** Cousins of Artificial Intelligence

(Source: <https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55>)

Deep learning is again a subset of machine learning mainly referred as deep artificial neural networks are a set of algorithms that have set a new records in terms of accuracy in many important areas of computer vision such as image recognition, sound recognition, recommender systems, natural language processing *etc.* Computer vision is concerned with the theory and technology for holding artificial systems which obtained information from images, videos multi-dimensional data *etc.* Deep neural network employs the mapping of input layer to the output layer over a series of stacked layers of nodes (Mao *et al.*, 2016). Most recently, deep convolutional network successfully applied in the area of object detection and classification and the results were out-performing than classical machine learning approaches in many diverse domains *e.g.*, pattern recognition (He *et al.*, 2016), instance detection and segmentation (Girshick, 2015), biomedical image segmentation (Ronneberger *et al.*, 2015) *etc.* A wide range of applications of deep learning are also available in the area of plant phenotyping *e.g.*, biotic stress identification in banana, pear, cherry and peach (Sladojevic *et al.*, 2016), apple scab and black-rot detection (Mohanty *et al.*, 2016), cassava brown streak disease detection (Ramcharan *et al.*, 2017) *etc.* Thus, it has been seen that, computer vision integrated with machine learning techniques achieved a great importance in the area of agriculture and allied sectors. These scenarios and importance of plant phenotyping are the main motivation in deciding thesis areas as well as the objectives.

## **1.5 Problem definition and Motivation**

In order to design rice and wheat genotypes with higher yield and greater stability under various constraints, the phenotype-genotype gap must be bridged up. Rice and wheat crops have been selected in this research work because of their importance as major food crops, availability of diverse germplasm resources, the need to enhance water use efficiency and low temperature stress tolerance and the vast available background knowledge of its physiology, genetics and genomics. Conventional measurement of the yield related traits like plant biomass and spike identification and counting through naked-eye in a large scale is very time consuming, labour intensive and destructive. Some literatures are available for estimating plant biomass through non-destructive image analysis technique (Paruelo

*et al.*, 2000; Mizoue and Masutani, 2003; Golzarian *et al.*, 2011; Schirrmann *et al.*, 2016). In most of the cases, projected shoot area computed from visual images (VIS) as a linear function of plant biomass. None of them, considered water content which is the key determinant of biomass (Seelig *et al.*, 2008). In this study, it is hypothesized that combined use of VIS and near infra-red (NIR) image can compute plant biomass more precisely than VIS image as NIR reflectance image is used to measure water content of the plant (Stenberg *et al.*, 2010; Fernández *et al.*, 2015).

Besides, spike or ear emergence is a critical phenological event in wheat development, as it is required for application of nitrogen, water and other critical inputs for crop production. Further, yield estimation in wheat has received a significant research attention as it is an important primary food for a large proportion of the world's population (Bognár *et al.*, 2017). Since spike number is a key factor that determines grain number per unit area and thus yield, counting of number of spikes/ears is an important measure to determine yield of the plant (Jin *et al.*, 2017). Therefore spike detection and counting is important for phenology based input management for crop production and assessing the crop yield. Counting of number of spikes per plant or per unit area through naked-eye is a laborious and time consuming process. Image analysis is being used in the area of spike/panicle detection and characterization (Bi *et al.*, 2010; Zhao and Ajay, 2015; Li *et al.*, 2017; Sadeghi-Tehran *et al.*, 2017; Pound *et al.*, 2017; Hasan *et al.* 2018). Bi *et al.* (2010) carried out, analysis by capturing images after cutting the individual spikes from the plant which is destructive procedure. Colour and texture component have been used in this study by Li *et al.* (2017) and to identify wheat spikes which is not totally machine depended as manual intervention is required to define the texture and colour intensity range for the segmentation purpose. In the recent trend (Li *et al.* 2017; Sadeghi-Tehran *et al.* 2017; Pound *et al.* 2017; Hasan *et al.* 2018), it has been seen that computer visions particularly object detection plays an important role in non-destructive plant phenotyping through digital image analysis which is being helpful for automatic detection and counting of spikes in wheat plant. Besides, we have high-throughput imaging facility (LemnaTec GmbH, Aachen, Germany) established at Nanaji Deshmukh Plant Phenomics Centre, ICAR-IARI, New Delhi, India to collect image dataset of large amount of wheat and rice plants. Based on these, the

objectives have been decided under the thesis entitled “**Image Analysis Algorithms for High-throughput Phenotyping of Rice and Wheat**”.

## **1.6 Objectives**

1. To develop methodologies for deriving phenotypic traits of rice and wheat from high throughput images based on computational algorithms.
2. To evaluate and compare the performance of the developed methodologies against existing methods.
3. To implement the developed methodologies in the form of software package for high throughput phenotyping.

## **1.7 Plan of thesis**

In this thesis, new methodologies have been developed based on non-destructive image analysis for deriving phenotypic traits like Leaf Fresh Weight (LFW) in rice plant and spike identification and counting in wheat plant. VIS and NIR imaging has been used for estimating LFW. We have developed image processing based algorithm to compute two image derived parameter *i.e.*, Green Leaf Proportion (GLP) from VIS image and Mean Gray Intensity (NIR\_MGI) from NIR image for estimating LFW. GLP is the proportion of green pixel area to the total leaf area whereas NIR\_MGI is the mean gray values of the projected leaves on the image. The image derived parameter *i.e.*, GLP from VIS image and NIR\_MGI from NIR images have been used for building the machine learning model to estimate LFW. The proposed approach is named as VN\_LFW. Artificial Neural Network (ANN) technique has been used in machine learning model development and its performance has been compared with linear regression technique. The ANN model has been developed by using GLP and NIR\_MGI as the input node in the input layer and the corresponding actual LFW as the output node in output layer. Distinctive combinations of hidden layer and hidden nodes in each layer has been attempted, out of which best fitted ANN model has been selected. The developed ANN model for LFW estimation has been compared with the conventional destructive method as well as other image processing based approaches like, linear function of projected shoot area and regression approach based on GLP and NIR\_MGI. The statistical

evaluating parameters Mean Absolute Percent Error (MAPE) and Root Mean Square Error (RMSE) has been used to compare the performance of the mentioned approaches of LFW estimation. Macro has been developed in Matlab software for LFW estimation. The macro consists of two parts. First part deals with GLP measurement from VIS image while second part deals with NIR\_MGI measurement from NIR image.

For spike identification in wheat plant, deep learning models have been developed namely **SpikeSegNet** (Spike Segmentation Network) and **LGspikeNet** (Local patch extraction and Global mask refinement Spike detection Network) based on convolutional encoder-decoder deep learning technique. For training the network, VIS images and its corresponding mask images containing class label (*i.e.*, spike region only) have been used. Output of the developed model is a mask image containing spikes region only. After identification of spikes, flood-fill image analysis based object counting technique has been applied on the output image of the developed network for counting spikes number per plant. Details of the developed methodologies for estimating LFW in rice plant and spikes identification and counting in wheat plant has been elaborated in chapter III. For evaluating the performance of the proposed approach of spike identification and counting, statistical parameter like precision, accuracy, robustness, Jaccard Index (JI) have been computed. Performance of the proposed model (*i.e.*, **SpikeSegNet** and **LGspikeNet**) has also been compared with the conventional destructive measurement. Details of the statistical parameter for evaluation are discussed in chapter III and the results are discussed in chapter IV. Web-based software for wheat spike identification and counting has also been designed and developed in this study. HTML, CSS and JavaScript has been used to build Client Side Interface Layer (CSIL) of the software architecture and Server Side Application Layer (SSAL) has been implemented using FLASK web development tool. TensorFlow, Keras deep learning framework and several python libraries like numpy, scipy, matplotlib *etc.* have also been used in SSAL to develop deep learning module for spike/ear identification in wheat plant. For counting spike number “*analyse particles*” function of imageJ is integrated with SSAL. Input of the software is VIS image of the wheat plant grown in pot culture and output is binary image/mask image consisting of spike regions and corresponding spike count.

The whole thesis is divided into five chapters:

**Chapter-I** of the thesis *i.e.*, the present chapter gives a brief introduction to non-destructive plant phenotyping, importance of image analysis and machine learning in plant phenotyping, definition of the problem, motivation and objectives of the study. This chapter also contains orientation plan of the thesis.

**Chapter-II, Review of Literature**, consists of brief reviews in the field of non-destructive plant phenotyping through digital image analysis, high-throughput platforms available for non-destructive plant phenotyping and application areas of deep-learning in agriculture and allied fields.

**Chapter-III** deals with the **Material and Methods** that are used in the development of the methodologies for deriving LFW in rice plant and spike identification and counting in wheat plant and as well as developing software for the same.

**Chapter-IV, Results and Discussion** on outcome of the proposed ANN based methodologies for deriving LFW in rice plant and deep learning based approach of spike identification and counting in wheat plant has been discussed here.

**Chapter-V** deals with **Summary and Conclusion** of the whole research work.

Followed by **Abstract** (in English and Hindi), **References**, **Appendix** and **Annexure**.

**Appendix** is further divided into two parts. **Appendix I** consists of Matlab (GLP.m) macro for GLP computation from VIS image and NIR\_MGI computation from NIR image (NIR\_MGI.m). **Appendix II** consists of python code for training, testing and performance evaluation of the developed encoder-decoder model, java code for spike counting and source code for web-based software for spike identification and counting.

**Annexure** is also further divided into two parts. **Annexure I** contains ground truth of biomass (LFW) data of 104 samples and **Annexure II** consists of wheat variety data of 200 plants.

## CHAPTER II

### REVIEW OF LITERATURE

---

#### 2.1 Introduction

Precise and accurate measurement of the plant phenotyping parameters in high-throughput and non-destructive manner plays an important role in the genetic improvement of the crop plants. Well accounts of articles are available in this field using digital image analysis. Digital image analysis involves the manipulation and interpretation of digital image with the aid of computer. In this chapter, a brief account of review of literature exists in the field of plant phenotyping parameter estimation concerned with yield related traits and application of deep-learning techniques in computer visions has been given. Review of high-throughput plant phenotyping platforms are also part of this chapter.

#### 2.2 Plant phenotyping through digital image analysis

Plant phenotyping through digital image analysis includes the development of computational algorithms to measure plant phenotypic parameters using non-invasive imaging technologies. Visual (or, RGB) and Near-infrared (NIR) non-invasive imaging technologies are used in yield related phenotypic traits quantification from last two decades. RGB images are used in measuring biomass, phenology (*i.e.*, spike/panicle identification especially in cereal plants) and leaf health (*i.e.*, leaf area, chlorophyll content) of the plant. NIR images are mainly used in quantifying moisture content presents (Serrano *et al.*, 2000) in the plants parts. The conventional technique of biomass measurement of the plant involved weighting of the plant parts after cutting. Few projects are available to estimate biomass of an individual plant by using digital image analysis technique. In most of the cases, projected shoot area of the plants captured on two dimensional images was used as a parameter to predict the plant biomass.

**Lukina *et al.* (1999)** conducted a study to predict percent vegetation coverage and biomass of winter wheat canopies growing in the field using digital image processing based on red-green-blue (RGB) color. The digital images were converted

from 8-bit RGB tagged image file format (TIFF) files to produce binary pseudo-color image. Percent of pixels corresponding to the vegetation was then calculated and considered as the percent vegetation coverage for each plot in the field. The results reflected that binary pseudo-color images provided useful estimates of percent vegetation coverage that were highly correlated with the ground truth.

**Montès *et al.* (2000)** proposed a method to measure biomass of agroforestry tree thuriferous juniper woodland (*Juniperus thurifera* L.) in High Central Atlas mountains (Morocco). This method was based on reconstructions of volumes of different components of the tree by using digital image processing. They had taken images from two orthogonal-views of different component of the tree. Then, using the volume and the density of each component of the tree, biomass of the whole tree was estimated. Regression curves were established and a second-order polynomial equation gave the best result to estimate the biomass with a high coefficient of determination ( $R^2 = 0.96$ ).

**Paruelo *et al.* (2000)** proposed a photographic method to estimate biomass in semiarid grasslands. This method was based on the relation between percentage of green pixels on the digital image and green biomass. The correlation of green pixels and green grass biomass was found as 0.87 ( $n=36$ ,  $p<0.001$ ) but in case of total plant biomass it was very lower (0.59).

**Smith *et al.* (2000)** proposed a digital photograph based method to assess quantities of live and dead plant biomass in the Florida Everglades. Images were captured within an open-sided frame. Then the images were transformed by using imaging software in such a way that live, dead and the absence of plant material were represented by the colors green, red, and black, respectively. Subsequent pixel counts were done and regression analyses showed strong correlations ( $R^2 > 0.84$ ) between estimated and actual dry weights.

**Lim and Treitz (2004)** proposed a conceptual model to describe the relation between laser heights metrics derived from airborne discrete laser scanner data with above ground biomass. In this conceptual model, they introduced the concept of canopy-based quantile estimators of the above ground forest biomass and it was applied on the uneven-aged, mature to over mature and tolerant hardwood forest.

Results obtained from the 0<sup>th</sup>, 25<sup>th</sup>, 50<sup>th</sup>, 75<sup>th</sup> and 100<sup>th</sup> percentiles of the distributions of laser canopy heights to estimate above ground biomass were reported. The coefficient of determination ( $R^2$ ) for each model was greater than 0.80 and any two models were differing at most up to 9%. Differences in RMSE between models for above ground total, stem wood, stem bark, live branch and foliage biomass were 8.1, 5.1, 2.9, 2.1 and 1.1 Mg ha<sup>-1</sup> respectively.

**Golzarian *et al.* (2011)** developed a model based on plant specific weight as a function of plant area and plant age to estimate the shoot dry weight. 320 plants of two bread wheat varieties were grown in hydroponics system for this purpose. Their proposed method showed very small difference between actual and estimated shoot biomass.

Leaf area is used for making decisions regarding cultivation pattern, trimming, pruning and managing fertilization schedules. Leaf area measurement is important in studying plant biological characteristics and in guiding agricultural production practice (Sestak and Catsky, 1971). Li-Cor 3100 leaf area meter is one of the most popular digital devices to measure leaf area. Another non-digital method involves the use of grid paper. After placing the leaves on the grid paper, the number of grid squares is calculated and area of the leaf is computed by multiplying the number of grid squares with area of one grid square. These methods are considered as ground truth of leaf area measurement but, are destructive, laborious and time-consuming. In this scenario, digital image analysis poses an alternative role.

**Li *et al.* (2008)** showed that measuring of leaf area based on image processing led to very high accuracy compared with the estimated leaf area using the grid paper method (non-digital method) by using a camera and personal computer. They used a wooded box and made a hole in the center of the top face to install the camera and placed it in the hole and fixed the distance between the lens of camera and bottom at 450 mm. The camera was adjusted to be vertical to the flat surface and the lighting conditions were controlled using natural light focused on the leaf under a camera to take photos.

**Chaohul *et al.* (2010)** used a non-destructive leaf area measurement through the use of Hough Transformation to acquire the coordinates of quadrangle corner points in

distorted image and thresholding for image segmentation. To eliminate the effect of holes in the leaf, contour extraction approach was used where pixel scanning from one side to opposite side was implemented in four directions to extract contour and leaf area was measured by pixel number statistic. They found absolute error 2.88.

**Marcon and Mariano (2011)** developed two models for leaf area measurement of coffee plants using digital image analysis. One was based on the height and width of the canopies and other based on the area of digital image of a tree. Firstly, the images were corrected by frequency histograms and then segmentation thresholding was done by using Otsu thresholding method. The results of the developed models were compared with real area of the leaves using digital scanner and adjusted  $R^2$  was found as 0.82 and 0.91 for model 1 and 2 respectively.

**Patil and Bodhe (2011)** described digital image processing techniques to calculate betel leaf area through an algorithm which was based on converting RGB (Red Green Blue) to grey scale image and grey scale image to binary image. With the implemented algorithm the relative error between area measurement by the proposed method and the actual value was 0.029.

Manual chlorophyll extraction procedure using DMSO (Dimethyl sulfoxide) method (Porra *et al.*, 1989) is accurate and considered as accurate but, it is destructive, laborious and relatively time consuming. In this context, color image analysis is a fast and automated alternative.

**Moghaddam *et al.* (2011)** applied machine vision approach for estimating chlorophyll content in sugar beet leaf. The results indicated that the neural network model trained with the RGB component was capable of proper estimation of chlorophyll level with  $R^2 = 0.94$ .

**Mahdi *et al.* (2012)** used following formula to non-linearly map the normalized value of Green content (G), with respect to Red (R) and Blue (B), using a logarithmic sigmoid transfer functions.

$$Ch\ value = \logsig\left(\frac{G - \frac{R}{3} - \frac{B}{3}}{255}\right)$$

The correlation between image processing technique based on the above formula and laboratory measured chlorophyll content for tomato, lettuce and broccoli as 0.968, 0.896 and 0.914 receptively.

**Misra *et al.* (2018)** did a comparative study of the existing techniques (Kawashima and Nakatani, 1998; Yuzhu *et al.*, 2011; Adamsen *et al.*, 1999 and Mahdi *et al.*, 2012) of chlorophyll content estimation through image processing on a real experimental data. The technique given by Adamsen *et al.* (1999) was found as best in their experiment with root mean square 0.67.

Some research works are available in the literature on wheat and rice crop in the area of computer vision to detect and characterize objects particularly spikes and panicles of the plant.

**Bi *et al.* (2010)** used image processing technique to know the information about growth status of wheat plant by measuring the spike characteristics, such as awn number, awn length and spike length *etc.* It provides the facilities of non-destructive measurement of wheat spike morphological characteristics and classification of several wheat species based on these characteristics. For this purpose, firstly, wheat awn was removed through the whole wheat plant and then images were taken. Then spike length was calculated by using the method of spindle direction angle and external rectangle length. Awn length and number of awn was calculated through the method of thinning and corner detection, and spike type was estimated through width coefficient proportion method. Secondly, a three layer back-propagation neural network was designed with the extracted characteristic parameters to classify the 240 pictures of 4 wheat varieties. The recognition accuracy was 88%.

**Li *et al.* (2017)** proposed an approach to detect and characterize the geometric properties of spikes of a single wheat plant grown in a controlled environmental condition. They have used color index method for plant segmentation and neural network method with Laws texture energy for spike identification with around 80% accuracy.

**Sadeghi-Tehran *et al.* (2017)** employed bag-of visual- words approach to identify growth stages in field grown wheat. They have used SIFT algorithm for low level features extraction and finally support vector machine classification technique was

used to classify growth stages. In flowering stage, accuracy was 85% and 99% in case of late growth stage.

**Pound *et al.* (2017)** presented a deep learning approach for localizing wheat spikes and spikelet with around 95% accuracy and the plants were imaged in a small-purpose built chamber with uniform background.

**Hasan *et al.* (2018)** used region based Convolutional Neural Network (R-CNN) approach for detecting, counting and analysing wheat spikes in the field condition. They captured images using RGB camera mounted on land based imaging platform. The average detection accuracy was ranging from 88 to 94% across different sets of test images.

**Duan *et al.* (2015)** proposed a method for counting rice panicle as maximum number of panicle obtained from 12 multi-angle images of the plant. After applying hysteresis thresholding for segmentation over  $i_2$  component of  $i_1i_2i_3$  colour space based on Karhunen-Loeve transformation, local region growing algorithm based on normalized RGB colour space was applied and then artificial neural network was used for pattern recognition. They achieved test accuracy of 94.27% for panicle region identification but only 54.3% for panicle number.

**Xiong *et al.* (2017)** proposed a segmentation method for rice panicles in the field named Panicle-SEG by using small window patch of size  $32 \times 32$ . The patches were centered on the weighted center of Simple Linear Iterative Clustering super pixel region. These weighted patches were used as input to convolutional neural network. Mask image generated using PhotoShop and original image was used to label patches. Caffe deep learning framework was used with update process based on stochastic gradient descent for CNN and named Panicle-SEG-CNN model. As the rectangle patches includes pixels which doesn't represent panicle, to remove these negative pixels from patches entropy rate super-pixel optimization was used.

**Hunt and Rock (1989)** and **Serrano *et al.* (2000)** used Near-Infrared (NIR) reflectance in determining water content of the plant parts. The details have been discussed as follows:

**Hunt and Rock (1989)** conducted a study for detecting the changes in leaf water content using Near-Infrared (NIR, 0.7–1.3  $\mu\text{m}$ ) and Middle-Infrared (MIR, 1.3–2.5  $\mu\text{m}$ ) reflectance. The first objective of their study was to test the ability of the Leaf Water Content Index (LWCI) to determine leaf Relative Water content (RWC) of different species with different leaf morphologies. The second objective was to determine how the Moisture Stress Index (MSI;  $\text{MIR} / \text{NIR}$ ) varies with RWC and the Equivalent Water Thickness (EWT). The results of the sensitivity analysis indicated that the reflectance at 1.6  $\mu\text{m}$  for two different RWC must be known for accurate prediction of unknown RWC; thus the LWCI was impractical for field applications. MSI was found linearly correlated to RWC with each species having a different regression equation.

**Serrano *et al.* (2000)** used infrared imaging spectrometer to assess Relative Water Content (RWC) at the landscape level. The Water Index (WI) and Normalized Difference Water Index (NDWI), reflectance indices were computed from NIR water absorption bands and were found as the best indicators of canopy RWC. In this study, stepwise multiple regression technique was applied which revealed that canopy structure explained 36% and 41% of the variation in WI and NDWI, respectively. The relationship between WI and the canopy RWC was improved significantly when data from plots with green vegetation cover was greater than 70% ( $r^2=0.88$ ,  $p<0.001$ ). These results indicated that WI and NDWI were sensitive to variations in canopy relative water content at the landscape scale.

### **2.3 Application of deep learning techniques in agriculture and allied sectors**

Recently, deep learning technique has been successfully applied in computer vision, especially, in the area of object detection and classification. The results were also promising than classical machine learning approaches in many diverse domains.

AlexNet architecture (**Krizhevsky *et al.*, 2012**) of convolutional deep-learning technique consisting of 5 convolutional layers, 3 fully connected layers with Relu activation function was successfully applied in apple scab and black rot detection.

GoogLeNet architecture (**Szegedy *et al.*, 2015**) consisting of inception module instead of simple convolution layer and each inception module containing 3\*3 conv, 5\*5 conv, 3\*3 max\_pooling and 1\*1 conv was applied in cassava streak disease detection

(Ramcharan *et al.*, 2017) and biotic stress detection in pear, cherry, peach (Sladojevic *et al.*, 2016).

**Ronneberger *et al.* (2015)** developed UNet deep learning architecture for bio-medical image segmentation. The architecture consists of two paths. First path is the contraction path (encoder) used to capture the context in the image. The encoder consists of stack of convolutional and max pooling layers. The second path is the symmetric expanding path (decoder) used to enable precise localization using transposed convolutions.

**Mohanty *et al.* (2016)** presented the application of deep convolution neural network to identify 26 diseases over 14 crop species using 54306 public dataset images. They used two different architectures of CNN viz., GoogLeNet and AlexNet using Caffe framework of deep learning on Python and achieved a top accuracy of 99.35%. This accuracy was achieved using transfer learning with 80% training data and 20% test data.

## **2.4 Plant phenotyping facility**

High-throughput plant phenotyping is evolving as an important area, which provides efficient data management, image analysis, and result visualization of large-scale plant phenotypic data sets.

**Klukas *et al.* (2014)** presented Integrated Analysis Platform (IAP), an open-source framework for high-throughput plant phenotyping. For validation of IAP, they performed an example experiment that contained 33 maize (*Zea mays* ‘Fernandez’) plants, which were grown for 9 weeks in an automated greenhouse with nondestructive imaging. Subsequently, the image data were subjected to automated analysis with the maize pipeline implemented in their system. They found that the computed digital volume and number of leaves correlate with their manually measured data in high accuracy up to 0.98 and 0.95, respectively.

**Knecht *et al.* (2016)** presented an open-source, flexible image-analysis framework, called Image Harvest (IH), for processing images originating from high-throughput plant phenotyping platforms. Image Harvest offers functionalities to extract digital traits from images to interpret plant architecture-related characteristics. To

demonstrate the applications of these digital traits, they had done their experiment on a rice (*Oryza sativa*) diversity panel and genome-wide association mapping had performed using digital traits of different plant ideotypes. They identified three major quantitative trait loci on rice chromosomes 4 and 6, which co-localize with quantitative trait loci known to regulate agronomically important traits in rice.

**LemnaTec** GmbH, is a company in Aachen, Germany that supplies software and automated research platforms for digital plant phenotyping through image analysis technique. The product includes *LemnaTec Scanalyzers* to provide a range of phenotyping research platforms from bench-top enclosures to systems that monitor large fields; *LemnaTec OS*, to support the plant phenotyping software system; and a range of supported *cameras and sensors* (RGB, NIR, Hyperspectral, and Fluorescence) to address every plant phenotyping requirement.

In this chapter, a brief account of previous work done in the field of plant phenotypic parameter estimation through image analysis has been discussed. Some high-throughput imaging platforms are also enlightened here. The next chapter of the thesis (chapter III) is concerned with the detailed discussion of materials and methodologies used in developing image analysis algorithm for measuring yield related traits namely Leaf Fresh Weight (LFW) in rice plant and spike identification and counting in wheat plant.

## CHAPTER III

### MATERIALS AND METHODS

---

#### 3.1 Introduction

In this study, new methodologies for deriving phenotyping traits in rice and wheat plant through digital image analysis have been proposed. The methodologies have been developed based on image processing and machine learning technique to estimate Leaf Fresh Weight (LFW) in rice plant and detection and counting of spikes/eras in wheat plant. Machine learning techniques further include artificial neural network (ANN) technique, deep learning techniques and its variants. Details of experimental materials and methodologies, tools and techniques used in this study are given in the following sections.

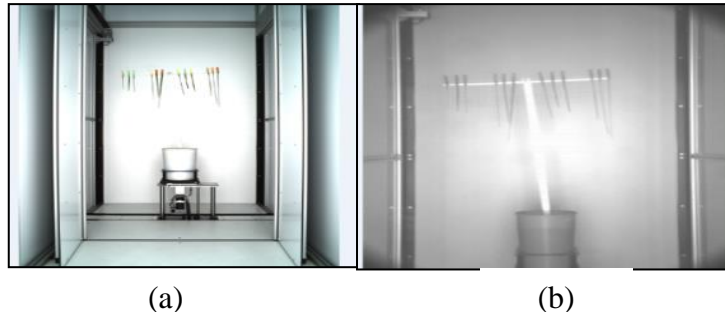
#### 3.2 Image analysis based Leaf Fresh Weight (LFW) estimation in rice plant

Leaf Fresh Weight (LFW) is used to estimate plant biomass which determines plant growth rate as well as net primary production. Several digital image processing based approaches are available to estimate fresh biomass of plants (Paruelo *et al.*, 2000; Mizoue and Masutani, 2003; Goltzarian *et al.*, 2011; Schirrmann *et al.*, 2016) but, majority are based on projected shoot area estimated from the visual (VIS) images only. These approaches do not consider the water content of the plant tissues which have a significant amount of contribution (about 70-80%) in fresh biomass (Paruelo *et al.*, 2000). Since water absorbs radiation in the near infra-red (NIR) (900 nm -1700 nm) region, it is hypothesized in this study that combined use of VIS and NIR imaging can predict the fresh biomass more accurately than the VIS image alone.

##### 3.2.1 Image acquisition for LFW estimation

Rice leaves from mini-core rice genotypes have been harvested and subjected to dehydration at room temperature to generate samples of rice leaves with different fresh mass. The leaves are then arranged in a hanger as shown in Fig 3.1. Leaf samples are imaged using VIS and NIR sensors (LemnaTec GmbH, Aachen, Germany) at Nanaji Deshmukh Plant Phenomics Centre, ICAR-IARI, New Delhi, India. Total 26 images have been taken and each image contains 4 set of leaves (*i.e.*,

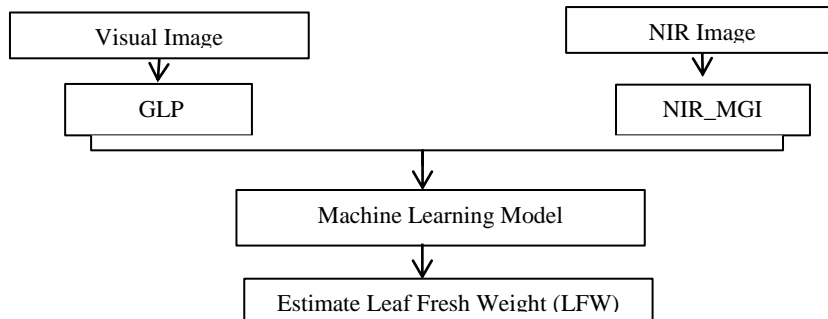
104 set). RGB camera of spectral response 400 to 700 nm with sensor (6576 x 4384 pixels) has been used to collect VIS images, whereas GoldEye P-032 SWIR camera of spectral response 900 to 1700 nm with InGas sensor (640\*480 pixel) has been used for taking NIR images. LFW is measured for each set with the help of weighing machine (Annexure 1) which is used as ground truth value to validate the proposed approach.



**Fig 3.1** (a) visual image, (b) NIR image

### 3.2.2 Proposed approach of LFW estimation - VN\_LFW

For estimating LFW, two image derived parameters *viz.* Green Leaf Proportion (GLP) from VIS image and Mean Gray Intensity (NIR\_MGI) from NIR images have been developed. As green leaf area has significant relation with the weight (Mora *et al.*, 2011) of the plant, hence, it is hypothesized that GLP can play a vital role in LFW estimation. These parameters (GLP & NIR\_MGI) are used as input for building machine learning model to estimate LFW (Fig 3.2) in rice plant. The proposed approach is named as **VN\_LFW**. ANN technique has been used in model development and its performance has been compared with linear regression technique.



**Fig 3.2** Flow diagram of **VN\_LFW**

GLP from VIS image and NIR\_MGI from NIR image have been obtained by using the following algorithms:

***Computation of GLP from VIS image:***

*Step 1:* Apply operation ‘O’ for background extraction and Otsu’s thresholding (Otsu, 1979) for background removal.

$$O = (G-R) / (G+R) \quad (i)$$

Where, G=Green content per pixel

R=Red content per pixel

*Step 2:* Multiply outcome image of *step 1* with the original image for extracting the original gray values.

*Step 3:* Extract that portion of the resulting image where  $G > B$  and  $G > R$ , it will be Green Leaf Area (GLA).

*Step 4:* Green Leaf Proportion (GLP) has been calculated by using the following formula:

$$GLP = (GLA/TA) \quad (ii)$$

Where, GLA= Green Leaf Area

TA= Total Leaf Area (total pixel area covered by leaf)

***Computation for NIR\_MGI from NIR image:***

For background subtraction and mean estimation of gray value of foreground from NIR image, an algorithm has been proposed by taking inspiration from NIR imaging pipeline of PlantCV (Gehan *et al.* 2017). High gray values represent high reflectance and indicate low water content, while low gray-scale values represent high absorption and high water content in the leaf (Subramaniam1 *et al.*, 1999). The algorithm is given below:

***Algorithm for Gray value estimation from NIR image:***

*Step 1:* Subtraction of original image (leaf image) from background image.

*Step 2:* Thresholding of the subtract image using Otsu’s thresholding algorithm (Otsu, 1979): In this step, pixels that have signal value less than the threshold value will be set to 0 (black) and else set to 1 (white).

*Step 3:* Image Sharpening: This step has been done for capturing maximum amount of object of interest (leaf) especially when the background pixel intensity is problematic (full of noise). For this purpose Second derivative Laplacian filter (Huertas & Medioni, 1986) is applied to the original image.

*Step 4:* Subtraction of the filtered image from the original image for increasing the contrast between object and background.

*Step 5:* Sobel filtering (Kanopoulos *et al.* 1988) of the original image has been done for highlighting more ambiguous boundaries within the image.

*Step 6:* Median blur filtering (Huang *et al.* 1979) has been applied on the Sobel filtered image to decrease the amount of noise present in that image.

*Step 7:* Add outcome of *step 3* and *step 6*.

*Step 8:* Thresholding of the output image using Otsu's thresholding algorithm.

*Step 9:* Erode the output image of *step 8* using 3\*3 filter.

*Step 10:* Add outcome image of *step 2* and *step 9*.

*Step 11:* Multiply outcome image of *step 10* and the original image for extracting the original gray values.

The algorithm has been implemented in MATLAB software (version 2015b, MathWork, Natick, MA).

### ***Artificial Neural Network***

Artificial neural networks (ANNs) model are considered as a class of generalized nonlinear model that are able to capture various nonlinear structures present in the data set. The main advantage of this model is that it does not require prior assumption of the data generating process, instead it is largely depend on characteristics of the data popularly known as data-driven approach. Feed forward multilayer network is the most popular for regression problem. This model is characterized by a network of three layers of simple processing units, the first layer is one input layer, the middle layer is the one or more hidden layer and the last layer is one output layer.

The relationship between the output ( $y$ ) and the inputs ( $x_1, x_2, \dots, x_p$ ) can be mathematically represented as follows:

$$y = f \left( \sum_{j=0}^q \omega_j g \left( \sum_{i=0}^p \omega_{ij} x_p \right) \right) \quad (iii)$$

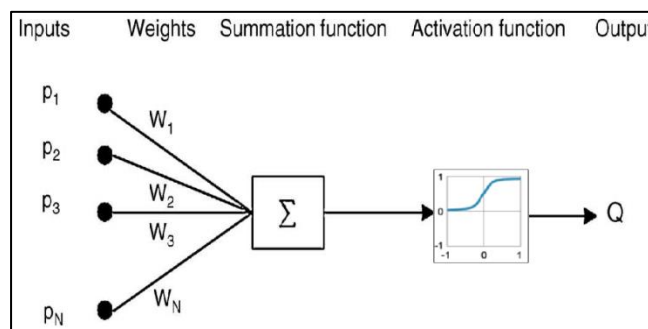
where,  $\omega_j (j = 0, 1, 2, \dots, q)$  and  $\omega_{ij} (i = 0, 1, 2, \dots, p, j = 0, 1, 2, \dots, q)$  are the model parameters often called the connection weights,  $p$  is the number of input nodes and  $q$  is the number of hidden nodes,  $g$  and  $f$  denote the activation functions at

hidden and output layer respectively. Activation function defines the relationship between inputs and outputs of a network in terms of degree of the non-linearity. Most commonly used activation functions (Table 3.1) are as follows:

**Table 3.1** List of activation functions

Activation function	Equation
Sigmoid	$\frac{1}{1 + e^{-x}}$
TanH	$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan	$\tan^{-1}(x)$

For regression problem generally sigmoid activation function is employed in hidden layer and identity activation function is employed in the output layer (Fig 3.3). The selection of appropriate number of hidden layers and hidden nodes in each layer is important in ANN modelling. Though there are no established theories available for the selection of hidden layer and nodes, hence experiments are often conducted for the determination of the number of hidden layer and hidden node in each layer. In this study, Gradient decent back propagation (Olabode & Olabode, 2012) has been employed for estimation of weight in neural network.



**Fig 3.3** Simple architecture of ANN

### ***Linear Regression***

Regression analysis is a statistical methodology which utilizes the relation between two or more quantitative variables where one variable can be predicted from the

other variable(s) (Seber & Lee, 2012). The functional relation between two variables can be expressed by mathematical formula. If X denotes the independent variable and Y denotes the dependent variable, the functional relation is of the form:

$$Y = f(X) \quad (iv)$$

Regression approach may be classified into two broad categories *viz.*, linear regression models and nonlinear regression models. The response variable (or, dependent variable) is generally related to other causal variables through some parameters. The models that are linear in these parameters are known as linear models; whereas in nonlinear models parameters appear nonlinearly. Linear models are considered as the most satisfactory approximations for most regression applications. Linear model can be stated as follows:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \quad (v)$$

Where,  $Y_i$  = value of the response variable in the  $i^{\text{th}}$  trial

$\beta_0, \beta_1$  = parameters

$X_i$  = value of the independent variable in the  $i^{\text{th}}$  trial

$\varepsilon_i$  = random error with mean zero and variance  $\sigma^2$

### 3.2.3 Model development and performance measurement of the proposed VN\_LFW approach

For developing the model, dataset (104 samples) has been divided into two parts randomly: 85% for training and 15% for testing. Feed-forward multilayer neural network (MLPNN) has been fitted using the NIR\_MGI from NIR image and GLP from VIS image as the input and the corresponding actual LFW as the output for the training data sets. As two independent variables (NIR\_MGI and GLP) and one dependent variable (LFW) are used in this study, the ANN architecture consists of one input layer with two input nodes and one output layer with one node. The choice of suitable number of hidden layer and hidden node within each layer is critical in ANN modelling. Despite the fact that there are no settled hypotheses accessible for the choice of hidden layer and hidden node, consequently tests are frequently used for the determination of the ideal estimations of hidden layer and

node. Hence, distinctive combinations of hidden layer and hidden nodes in each layer has been attempted, out of which best fitted ANN model is selected. “*Neuralnet*” package of R software (Teodoro, 2015) has been employed for the model development.

Performance of the proposed ANN approach of LFW estimation has been judged and compared by the existing approaches *i.e.*, based on linear function of projected shoot area as well as Linear regression approach based on GLP from VIS image and NIR\_MGI from NIR image. Mean Absolute Percent Error (MAPE) and Root Mean Square Error (RMSE) are measured to compare the performances of the approaches. The model with less MAPE and RMSE is preferred for prediction purposes. The MAPE and RMSE are computed as follows:

$$MAPE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| / y_i \times 100 \quad (vi)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (vii)$$

Where, n = total number of observations.

$y_i$  = actual value of observation  $i$  (LFW ground truth)

$\hat{y}_i$  = estimated value (predicted LFW).

### 3.2.4 Tools used in implementing VN\_LFW approach of LFW estimation

MATLAB software has been used to compute GLP from VIS image and NIR\_MGI from NIR image to estimate LFW in rice plant. MATLAB [The MathWorks Inc. MATLAB 7.0 (R14SP2). The MathWorks Inc., 2005] is a high-performance language which facilitates computation, visualization, and programming environment. Furthermore, it provides a modern programming language environment having sophisticated data structures, built-in editing and debugging tools, and also supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research. Various functions of “*Image analysis*” module (Table 3.2) of MATLAB have been used for this purpose as follows:

**Table 3.2** Matlab functions and their functionalities

<b>Functions</b>	<b>Functionalities</b>
<i>greythresh()</i>	Image thresholding using Otsu's method
<i>fspecial('laplacian')</i>	For applying Laplacian filter on image
<i>edge(image, 'sobel')</i>	For applying Sobel filter on image for edge detection
<i>midfilter2(image)</i>	For applying median filter on image for noise removal
<i>imerode(image,strel('square',2))</i>	Image erosion

“*neuralnet*” package of R software has been used for implementing ANN to estimate LFW in rice plant. Various functions (Table 3.3) of “*neuralnet*” package that has been used in this study are listed as follows:

**Table 3.3** R functions and their functionalities

<b>Functions</b>	<b>Functionalities</b>
<i>compute()</i>	Used in computation of the calculated network
<i>confidence.interval()</i>	Used in calculation of a confidence interval for the weights.
<i>prediction ()</i>	For calculation of a prediction
<i>plot.nn ()</i>	For plotting of the neural network

### 3.3 Spike identification and counting in wheat plant through digital image analysis

Detecting and counting of number of spikes per plant or per unit area through naked-eye is a laborious, error- prone and time consuming process. In this study, a new approach has been presented based on combined effort of digital image analysis and deep learning technique.

### 3.3.1. Image acquisition for wheat plants

In the Phenomics facility, wheat plants are grown in pots in the climate controlled greenhouse. Names of the genotypes used in this study are given in Annexure 2. Images of the plants are taken (Fig 3.4) by using 6576 x 4384 pixel RGB camera (LemnaTec GmbH, Aachen, Germany). It is hypothesized that images from one side cannot cover all the spikes of the plant; hence images from three different sides ( $0^\circ$ ,  $120^\circ$ ,  $240^\circ$ ) of plants are recorded by using the automated turning and lifting unit present inside the imaging unit. A uniform background has been maintained to increase the accuracy of separate background and plant regions. Images have been stored in lossless PNG format as facilitated by the LemnaTec facility. Imaging has been done during reproductive stage of the plant. After imaging, number of spikes per plant is counted manually which has been used as ground truth value to validate the proposed approach.



**Fig 3.4** Visual image of wheat plant

### 3.3.2 Proposed approach of spike identification and counting

The proposed approach involves identification and counting of spikes from the digital images of whole wheat plant. Spike identification through image analysis is a pixel wise segmentation problem of object (here, spikes). Convolutional encoder-decoder deep learning technique (<https://medium.com/@aggirma/part-1-convolutional-neural-network-in-a-nutshell-89f81a329ec3>) based model plays a promising role in this aspect [Ronneberger *et al.* (2015); Szegedy *et al.* (2015); Mohanty *et al.* (2016); Jaswal *et al.* (2019)]. In this study, two deep learning models have been developed for spike identification namely **SpikeSegNet** (Spike Segmentation Network) and **LGspikeNet** (Local patch extraction and Global mask

refinement spike detection Network) based on convolutional encoder-decoder deep learning technique. For counting the number of spikes per plant, “analyse particles” function of imageJ (Schneider *et al.*, 2012) software has been applied on the output image of the encoder-decoder model which implements flood-fill image analysis technique (Asundi and Wensen 1998).

### ***Convolutional Neural Network***

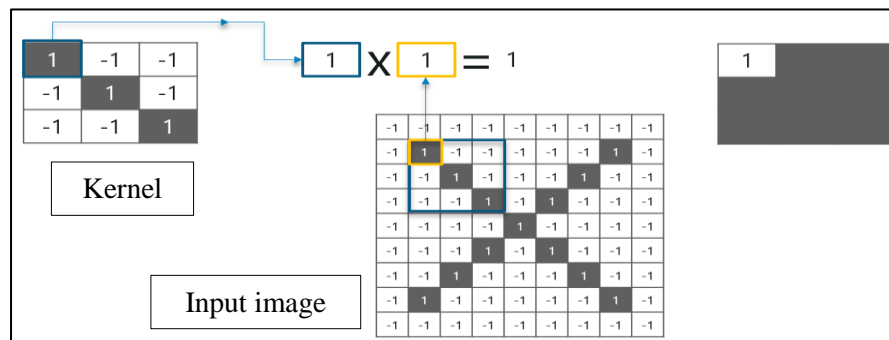
Convolutional Neural Network (CNN) is most commonly applied in analysing visual imagery ([https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)). CNN is a regularized version of multilayer perceptron (MLP). MLP usually refers to fully connected network where each neuron in one layer is connected to all neurons in the next layer. This "fully-connectedness" property makes the network prone to over-fitting of data. Regularization technique includes some form of magnitude measurement of weights to the loss function. Through regularization, CNN take advantages of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Besides, CNN involves relatively little pre-processing of the images compared to other image classification algorithms as the network learns through the filters/kernels (convolutions) where traditional algorithms are only hand-engineered. The independence of prior knowledge and human effort in feature design creates major advantage of this technique. A CNN network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of CNN typically consist of four (<https://medium.com/@aggirma/part-1-convolutional-neural-network-in-a-nutshell-89f81a329ec3>) layers:

1. Convolution layer
2. ReLU layer
3. Pooling layer and
4. Fully Connected Layer

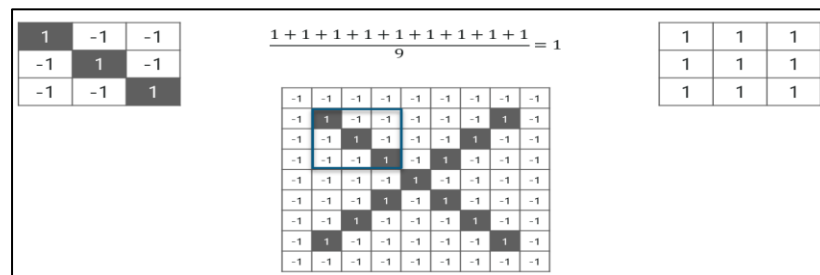
***Convolution Layer:*** Convolution refers to the mathematical combination of two functions to produce a third function ([https://en.wikipedia.org/wiki/Convolution\\_neural\\_network](https://en.wikipedia.org/wiki/Convolution_neural_network)). Convolution is performed on the input data by using filter or kernel to produce a feature map. For extracting the positional relationship in the image data, the kernel takes nearby pixels together and covers the input (image) by sliding the filter/kernel over it. At every

location, the kernel performs matrix multiplication and then sums the result onto the feature map. So, the convolution operation is performed in 4 steps:

- Line up the kernel window and the input image
- Multiply each pixel of the image by corresponding feature pixel in the kernel
- Add the values and find the sum
- Divide the sum by the total number of pixels in the feature kernel

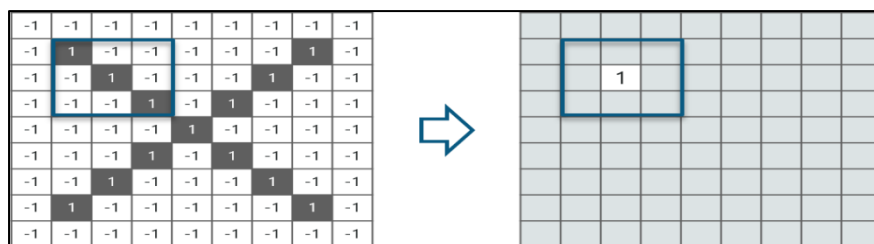


**Fig 3.5** First two steps of convolution operation



**Fig 3.6** Last two steps of convolution operation

Lastly final value obtained is placed at the centre of the filtered image (Fig 3.7).



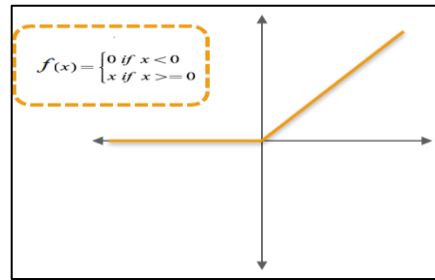
**Fig 3.7** Final value after completion of 4 steps of convolution operation is placed at the centre of the filter/kernel window

Now, the kernel moves around and does the same at every pixel in the image and the ultimate output obtained as follows:

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77


**Fig 3.8** The output features after completion of convolution operation on the input image of CNN deep-network

**ReLU Layer:** Rectified Linear Unit (ReLU) activation function ([https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)) only activates a node if the input is above or below a certain quantity (threshold). If the input value is below zero, the output is zero, but when the input rises above this threshold, it will have a linear relationship with the dependent variable.



**Fig 3.9** ReLU activation function

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



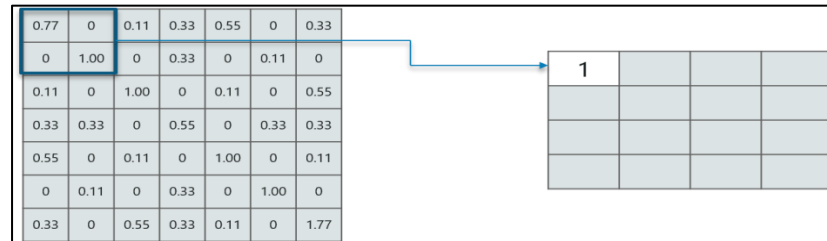
0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

**Fig 3.10** The output features after completion of ReLU operation

**Pooling Layer:** Output from the previous (ReLU layer) layers is smoothened for reducing the sensitivity of filters/kernels to noise and variation. Pooling operation ([https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)) is also known as subsampling and can be achieved by taking averages (average pooling) or taking the

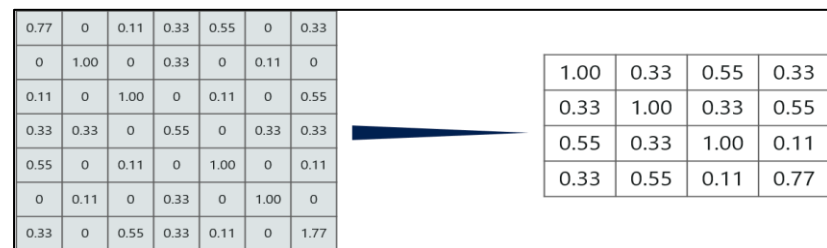
maximum (max pooling) over a sample of the signal. It is implemented by using the following steps:

- Picking up a window size (usually 2 or 3)
- Striding or moving the window down or across over the entire image and picking the maximum or average value from each window



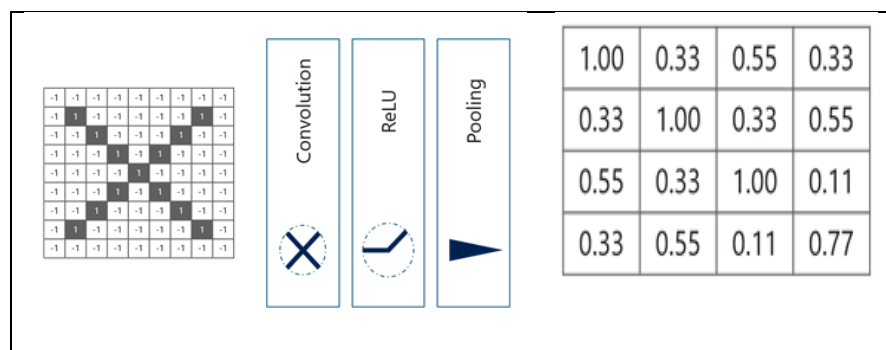
**Fig 3.11** Max-pool operation

For window size 2 (Fig 3.11), there are 4 values and the maximum value is 1 so pick 1. Also, it has been noted that a  $7 \times 7$  matrix came down to  $4 \times 4$  matrix after pooling operation (Fig 3.12).



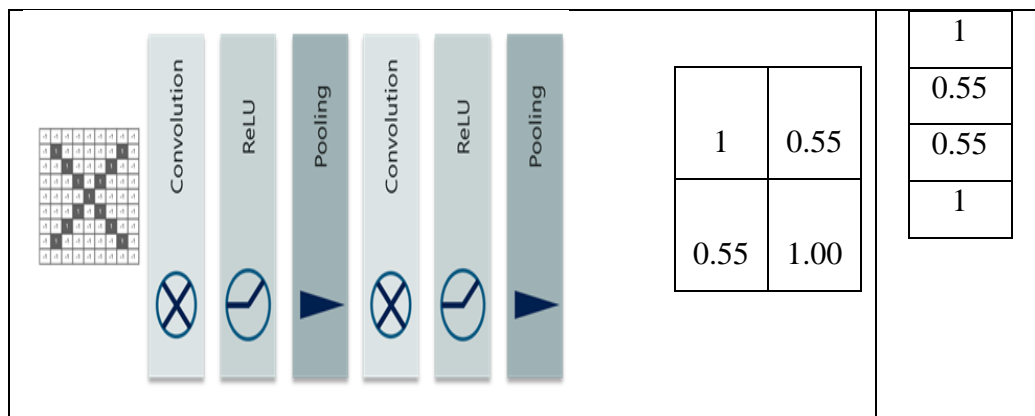
**Fig 3.12** The output features after completion of Max-pooling operation of CNN

So after passing through 3 layers (**Convolution, ReLU and Pooling**)  $7 \times 7$  matrix converted to  $4 \times 4$  matrix as shown below (Fig 3.13):



**Fig 3.13** Output features after completion of Convolution, ReLU and Pooling operation of CNN

**Fully Connected Layer:** For further reducing the size from 4×4 matrix, the 3 consecutive operations (Convolution, ReLU and Pooling) have to be performed iteratively (Fig 3.14). In fully connected layer ([https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)), neurons of preceding layers are connected to every neuron in subsequent layers which mimics high level reasoning to cover all possible pathways from the input to output. Fully connected layer is also the final layer where the classification actually happens. It can be viewed as the final learning phase where extracted visual features are mapped to the desired outputs and usually adaptive to classification tasks. Here, the output is a vector, which is then passed through softmax (Jang *et al.* 2016) to represent confidence of classification.



**Fig 3.14** Output features after completing 2 iterations of 3 consecutive operations of CNN (Convolution, ReLU, and Pooling)

From the output of fully connected layer, it can be observed that the value of 1<sup>st</sup> and 4<sup>th</sup> pixel is higher than 2<sup>nd</sup> and 4<sup>th</sup>. Therefore, if one input is come with fully connected layer output as

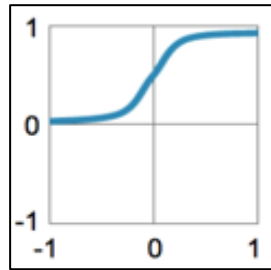
0.85
0.25
0.48
0.94

then, it will classified in this particular class, otherwise, in the different classes.

### Softmax Function

Softmax function (Jang *et al.* 2016) is a type of sigmoid activation function which is very useful in handling classification problems. It is non-linear in nature and usually used to handle multiple classes. The softmax function would squeeze the

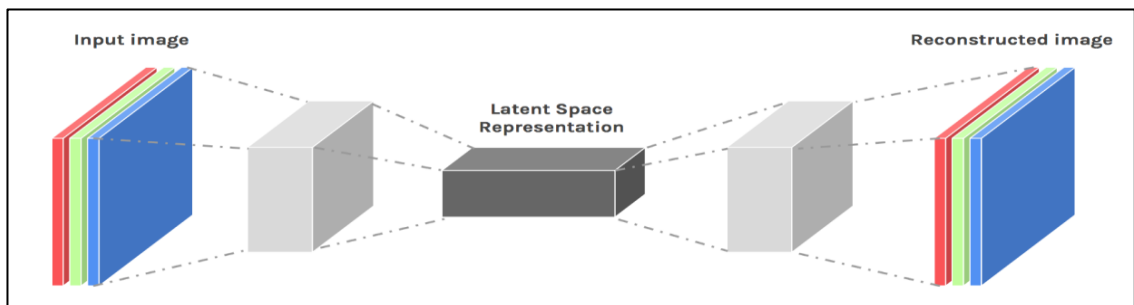
outputs for each class between 0 and 1 and would also divide by the sum of the outputs.



**Fig 3.15** Sigmoid activation function  $y=1/(1+e^x)$

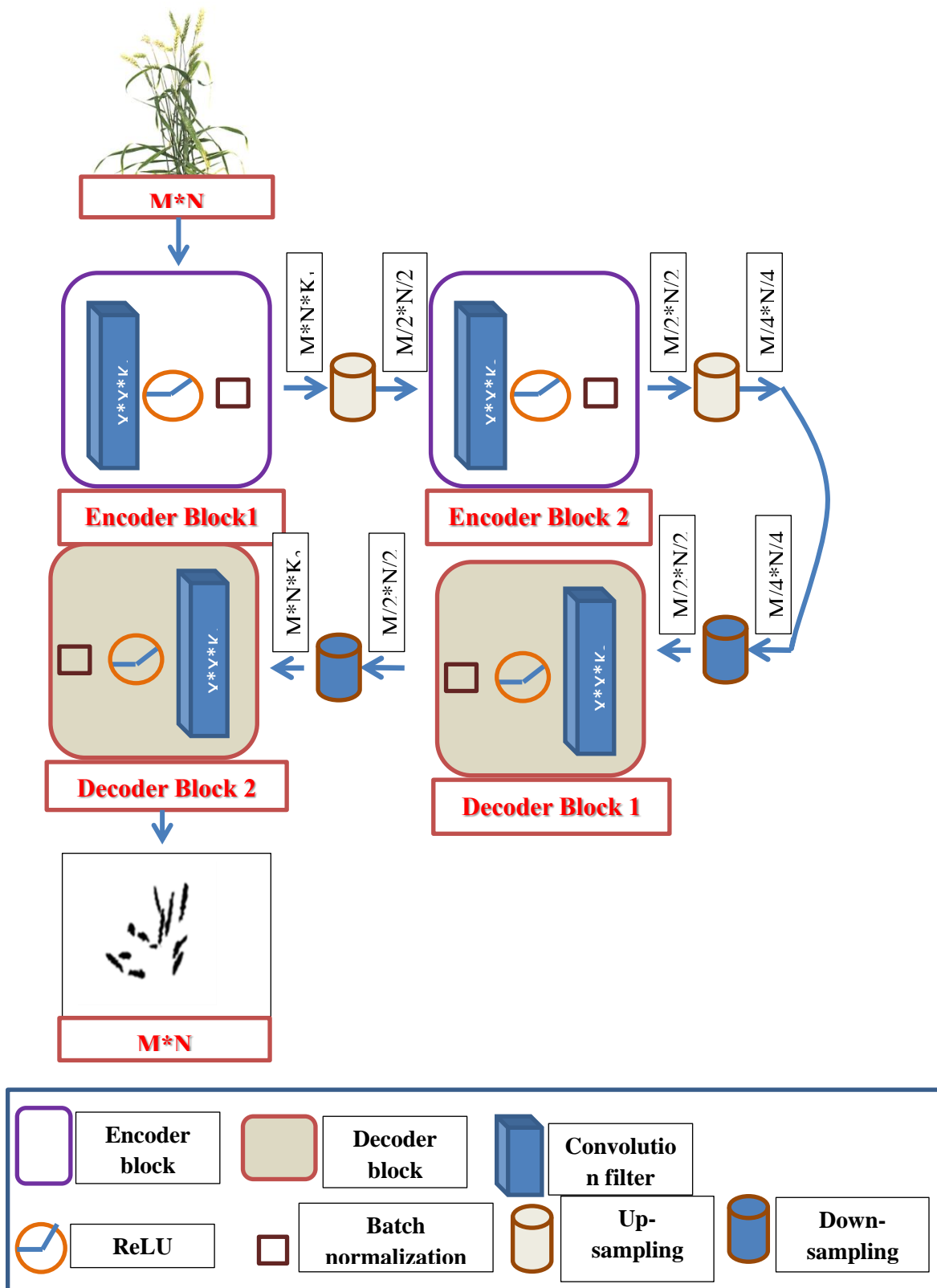
### ***Convolutional encoder-decoder deep network***

The convolutional encoder-decoder is a variant of CNN deep-network architecture based on encoding/decoding structure (Fig 3.16). Encoding network maps the raw input to feature representations holding the contextual information, while decoding network takes these feature representations as input, process it and produce corresponding segmentation mask as output (Mao *et al.*, 2016). The network is mainly used for pixel wise segmentation of objects. In this technique, the input is compressed into a latent-space representation, and reconstructed the output from this representation (<https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694>).



**Fig 3.16** Simple convolutional encoder-decoder deep-network architecture

A simple convolutional encoder – decoder deep network with 2 encoders and 2 decoders block has been explained here (Fig 3.17).



**Fig 3.17** Simple convolutional encoder – decoder deep network with 2 encoders and 2 decoders block

Generally, each encoder and decoder block consists of convolution layer and ReLU layer and additionally batch normalization (Ioffe and Szegedy, 2015) has been done to improve the performance and stability of network. Output of the first encoder block is sub-sampled/down sampled using max-pooling operation and the output is fetched to the second encoder block for further encoding in similar manner. In decoding technique, the spatial information from the encoder block has been forwarded to the decoder block along with incoming up-sampled features which helps in generating more location specific objects. Each decoder block consists of convolution, ReLU, batch normalization operations and additionally up-sampling/up-convolution is done as opposite to max-pooling operation to regenerate the features. Output of the final decoder will go through optimizer to update the weight depending on the loss function used. In deep-learning, several loss functions and optimizers are used based on the situations. Let's take an example image of size  $M*N$  and the inner functionality of each encoder and decoder blocks are explained as below:

**Table 3.4** Details of each encoder block and corresponding up-sampling features

Encoder Block #	Input to encoder block	Convolution filter/kernel size	Number of kernels	Output of encoder block	Input to max-pool (usually 2*2 window)	Output to max-pool
Block 1	$M*N$ (original image)	$x*x$ (usually, $x=3$ or $5$ )	$K_1$	$M*N* K_1$	$M*N* K_1$	$M/2*N/2* K_1$
Block 2	$M/2*N/2* K_1$	$x*x$	$K_2$	$M/2*N/2* K_2$	$M/2*N/2* K_2$	$M/4*N/4* K_2$

**Table 3.5** Details of each decoder block and corresponding down-sampling features

Decoder Block #	Input to transpose convolution	Output of transpose convolution	Input to decoder block	Convolution filter/kernel size	Number of convolution filters	Output of decoder block
Block 1	$M/4*N/4* K_2$	$M/2*N/2* K_2$	$M/2*N/2* K_2$	$x*x$	$K_2$	$M/2*N/2* K_2$
Block 2	$M/2*N/2* K_2$	$M*N* K_2$	$M*N* K_2$	$x*x$	$K_1$	$M*N* K_1$

### **Batch normalization**

Batch Normalization (BN) is a special normalization technique for neural networks ([https://github.com/aleju/papers/blob/master/neural-nets/Batch\\_Normalization.md](https://github.com/aleju/papers/blob/master/neural-nets/Batch_Normalization.md)).

As the inputs to each layer in neural networks depend on the outputs of all previous layers, the distributions of these outputs can change during the training process. This mechanism slows down the training process as each layer learns to adapt new distribution in every training step (<https://towardsdatascience.com/batch-normalization-theory-and-how-to-use-it-with-tensorflow-1892ca0173ad>). This problem is popularly known as internal covariate shift. In this context, batch normalization is used to normalize the inputs of each layer. During training time, batch normalization layer does the following:

1. Calculate the mean ( $\mu_B$ ) and variance ( $\sigma_B^2$ ) of the layers input ( $x_i$ ) consists of m samples.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad (viii)$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (ix)$$

2. Normalize the layer inputs using the previously calculated batch statistics.

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2}} \quad (x)$$

3. Scale and shift in order to obtain the output of the layer.

$$y_i = \gamma \bar{x}_i + \beta \quad (xi)$$

$\gamma$  and  $\beta$  are the distribution learned during training along with the original parameters of the network. So, if each batch had m samples and there are i batches:

## ***Loss function***

Machine learns by means of a loss function (<https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23>). It depends on how much the predicted results deviate from the actual results. Loss function learns to reduce the error in prediction with the help of optimization function. Loss functions can be classified into two categories depending upon the type of learning task - **Regression losses** and **Classification losses**.

**1. Regression losses:** It involves predicting a real-valued quantity. Following metrics are used for measuring regression losses:

- 1) **Mean Square Error (MSE):** MSE is only concerned with the average magnitude of error irrespective of their direction. In this technique, predicted values which are far away from actual values are penalized due to squaring which leads to less deviated predictions.

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (xii)$$

Where, n = total number of observations.

$y_i$  = actual value of observation i

$\hat{y}_i$  = estimated/predicted value

- 2) **Mean Absolute Error (MAE):** MAE is measured by taking average of sum of absolute differences between predicted and actual observations. Like MSE, it measures magnitude of error without considering their direction. MAE is robust to outliers as it does not make use of square.

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (xiii)$$

Where, n = total number of observations.

$y_i$  = actual value of observation i

$\hat{y}_i$  = estimated/predicted value

- 3) **Mean Bias Error (MBE):** This is same as MSE with the only difference that absolute values are not considered here. Although it is less accurate in practice, it can determine if the model has positive or negative bias.

$$MBE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)}{n} \quad (xiv)$$

Where, n = total number of observations.

$y_i$  = actual value of observation i

$\hat{y}_i$  = estimated/predicted value

**2. Classification losses:** It predicts output from a set of finite categorical values. Mainly cross entropy loss metrics are used for measuring binary as well as multi-class classification losses.

- **Cross Entropy Loss:** This is the most commonly used metric for classification problems. Value of the cross-entropy loss increases as the predicted probability value diverges from the actual label. An important aspect cross entropy loss is that it penalizes heavily the predictions that are confident but wrong.

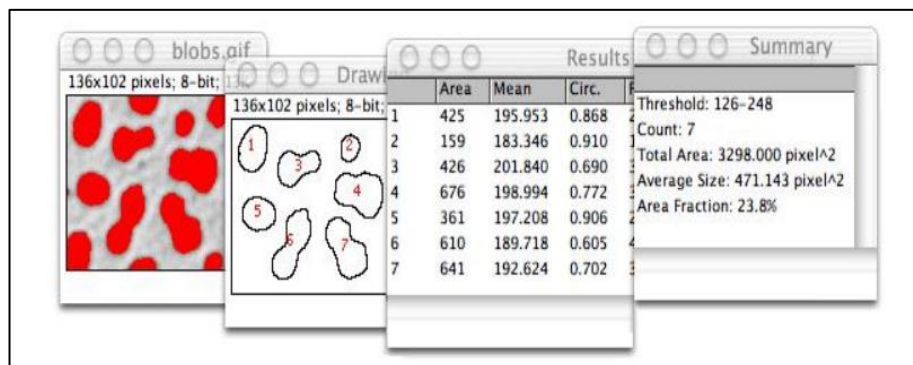
$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (xv)$$

### ***Optimizer***

The goal of machine learning or deep learning algorithm is to reduce the difference between the predicted output and the actual output. This is also called as a **Cost function** or **Loss function**. So the goal is to minimize the cost function by finding the optimized value for weights in the intermediate layers of the neural network and update them (<https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3>). For this purpose, several optimizers are available and all are variants of **Gradient descent** approach. Popular optimizers used in deep-learning are **SGD** (Stochastic Gradient Descent), **Adadelta**, **RMSprop**, **Adam**, etc.

### ***Proposed methodology of counting number of spikes in a single plant***

For counting the number of spikes per plant, “analyse particles” function of imageJ (Schneider *et al.*, 2012) software has been applied on the output image of the encoder-decoder model. The “analyse particles” function of imageJ implements flood-fill image analysis technique (Asundi and Wensen ,1998). Flood-fill technique employs object count by growing through similar pixel regions from the starting pixel. The “analyse particles” function counts and measures (pixel area) objects in binary or threshold images. It works by scanning the image until it finds the edge of an object and then outlines the object. After that the outlined objects are filled by using flood-fill approach and then it resumes the scanning process until reaches the end of image or selection (Fig 3.18).



**Fig 3.18** Output of “analyse particles” function of imageJ (source: <https://imagej.nih.gov/ij/docs>)

### **3.3.3 Model development and performance measurement of the proposed approach of spike identification and counting**

To develop encoder decoder model, image dataset of 200 plants with 3 images each are split into training and validation at 85:15 ratio randomly. RGB images and its corresponding ground truth (*i.e.*, mask image) images which consist of spike regions only are used in developing the model. Training has been done for 200 epochs. For each epoch, average accuracy and loss in the training and validation dataset has been computed based on the cross entropy loss function. Then, the developed model has been validated on the randomly selected validation dataset (images 15% of plants) to identify/detect spikes of the plant.

### ***Performance measurement of the proposed approach of spike identification***

For measuring the segmentation performance of the proposed network to identify/detect spikes, the validation dataset has been passed through the training network to extract the binary masks. The resulting binary masks have been compared with the ground truth masks and evaluated by difference performance parameter (Proenca *et al.*, 2010; Haindl & Krupička, 2015; Zhao & Ajay, 2015) as described below:

**Type I Error ( $E_1$ ):** For any  $r^{\text{th}}$  test image, exclusive-OR ( $\oplus$ ) operation is done to compute pixel-wise classification error ( $Pix\_Err_r$ ) between the predicted image ( $I^{\text{pred}}$ ) and corresponding ground-truth mask image ( $I^{\text{grtr}}$ ) of size  $p \times q$ ,

$$Pix\_Err_r \left( I^{\text{pred}}, I^{\text{grtr}} \right) = \frac{1}{p * q} \sum_{l=1}^q \sum_{k=1}^p [I^{\text{pred}}(k, l) \oplus I^{\text{grtr}}(k, l)] \quad (xvi)$$

Overall mean segmentation error or Type I error  $E_1$  has been computed by averaging the  $Pix\_Err_r$  of all the validation images:

$$E_1 = \frac{1}{N} \sum_{r=1}^N Pix\_Err_r \quad (xvii)$$

Where,  $N$  = no. of validation images.  $E_1$  lies within  $[0, 1]$ . If the value of  $E_1$  is close to “0”, it refers minimum error, whereas if  $E_1$  is close to “1”, it signifies large error.

**Type II error ( $E_2$ ):** For any  $r^{\text{th}}$  test image, the error rate  $E_2^r$  is computed by the average of false-positives (FPR) and false negatives (FNR) rates at the pixel level defined as:

$$E_2^r = 0.5 * FPR + 0.5 * FNR \quad (xviii)$$

Where,

$$FPR = \frac{1}{p * q} \sum_{l=1}^q \sum_{k=1}^p [(I^{\text{grtr}}(k, l) * I^{\text{pred}}(k, l)) \oplus I^{\text{pred}}(k, l)]$$

$$FNR = \frac{1}{p * q} \sum_{l=1}^q \sum_{k=1}^p [(I^{\text{grtr}}(k, l) * I^{\text{pred}}(k, l)) \oplus I^{\text{grtr}}(k, l)]$$

$E_2$  error rate is computed by taking the average errors of all the input images as given below:

$$E_2 = \frac{1}{N} \sum_{r=1}^N E_2^r \quad (xix)$$

**Jaccard Index (JI):** JI is measured by computing intersection over union (IoU) of the labelled segments for each class (*i.e.*, spikes and non-spikes) and thereafter averages of them are calculated as given below:

$$JI = \frac{1}{C} \sum_{i=1}^C \frac{C_{gg}}{Gr_g + Pred_g - C_{gg}} \quad (xx)$$

Where, C (=2) is the number of classes *i.e.*, spikes or and non-spikes

$C_{ii}$  = number of pixels in the image having ground truth label g and whose prediction is also g.

$Gr_g$  = total number of pixels in the image with ground truth label g.

$Pred_g$  = total number of pixels in the image whose prediction is g.

Similar to the above metrics, the final Jaccard Index is given by taking the mean of the Jaccard Index of all input test images.

Following performance parameters have also been used for measuring the segmentation performance of the proposed network to identify/detect spikes as follows:

- True positive (TP): # pixels correctly classified as spikes pixels.
- True Negative (TN): # pixels correctly classified as non-spikes pixels.
- False Positive (FP): # non-spikes pixels classified as spikes pixels.
- False Negative (FN): # spikes pixels classified as non- spikes pixels.

Then Precision, Recall, F-measure and Accuracy can be defined as:

- $Precision = \frac{TP}{TP + FP}$  measures, % of detected pixels are actually spikes (xxi)
- $Recall = \frac{TP}{TP + FN}$  measures, % of actually spikes spike pixels are detected (xxii)
- $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$  measures performance of the approach (xxiii)  
model

- $F_1 \text{ Score} = 2 \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$  measures robustness of the proposed network (xxiv)  
in detecting or identifying spikes

### ***Performance measurement in counting number of spikes***

For counting number of spikes per plant, the proposed approach of spike counting has been applied on the validation dataset (15%). Output (*i.e.*, spike count) of the proposed approach has been compared with the ground truth value *i.e.* spike per plant counted manually. The performance of the developed model in counting number of spike has been evaluated on the basis of precision, recall, accuracy and the  $F_1$  score based on true positive (TP), false positive (FP), true negative (TN), and false negative (FN) which are defined as follows:

- $TP$  = no. of objects correctly classify as spike
- $FP$  = no. of objects incorrectly classify as spike (*i.e.*, leaf, background) or overlapping spikes (connected objects)
- $FN$  = no. of actual spikes that are not visible in any of the side image
- $TN$  = is always 'zero' in this binary classification problem as background is not determined for object detection

Precision, recall, accuracy and  $F_1$  score are measured by using the above equations [(xxi), (xxii), (xxiii), (xxiv)] and are defined as follows:

- Precision: It measures the % of detected objects which are actually spikes
- Recall: It depicts the % of actually spikes have been detected among the ground truth
- Accuracy: It measures performance of the proposed approach
- $F_1$  score: measures robustness of the approach

### **3.3.4 Tools used in implementing the proposed methodology of Spike identification and counting**

The encoder-decoder deep learning architecture used in detecting spikes in wheat plant has been implemented through tensorflow and keras framework in python whereas for counting number of spikes imageJ API has been used.

**TensorFlow:**

TensorFlow is an open source machine learning framework used for implementing machine learning and deep learning applications (<https://www.tutorialspoint.com/tensorflow>). Google team has created TensorFlow to develop and research on fascinating ideas in the area of artificial intelligence. It is designed in Python programming language. Some important features of TensorFlow are as follows:

- It provides facility to define, optimize and calculate mathematical expressions with the help of multi-dimensional arrays called tensors.
- It includes programming facilities of deep neural networks and machine learning techniques.
- It facilitates a high scalable feature of computation with different data sets.

**Keras:**

Keras is a high-level Python library run on top of TensorFlow framework ([https://www.tutorialspoint.com/tensorflow/tensorflow\\_keras](https://www.tutorialspoint.com/tensorflow/tensorflow_keras)). It is developed with focus on understanding the deep learning techniques by providing the facilities of creating and maintaining the layers, it's shapes and mathematical details of the deep neural networks. It consists of several libraries to provide the following facilities to create a deep learning model:

- Loading the data
- Preprocess the loaded data
- Definition of model
- Compiling the model
- Fit the specified model
- Evaluate it
- Make the required predictions
- Save the model

Various modules of keras framework used in this thesis as follows:

**Table 3.6** Keras modules and their functionalities

Functions	Functionalities
<i>keras.models import Sequential</i>	For developing feed-forward CNN model of sequential type
<i>keras.layers import Dense, Dropout, Activation, Flatten</i>	For importing ‘core’ layers from keras that are used in almost any Neural Network
<i>keras.layers import Convolution2D, MaxPooling2D</i>	For implementing convolution layers
<i>keras.utils import np_utils</i>	For implementing some utilities to transform data
<i>model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])</i>	For compiling the deep-learning model, optimizer and loss functions are required.  <b>binary_crossentropy</b> loss function is used in binary classification (spikes or, non-spikes)  Some of the most popular optimization algorithms are: Stochastic Gradient Descent (SGD), ADAM and RMSprop.
<i>model.fit()</i>	Used in fitting the model

**ImageJ:**

ImageJ is a public domain Java image processing program [<http://rsbweb.nih.gov/ij/docs/>] inspired by NIH Image for the Macintosh. It runs, either as an online applet or as a downloadable application, on any computer with a Java 1.1 or later virtual machine. Downloadable distributions are available for Windows, Mac OS, Mac OS X and Linux. It can display, edit, analyze, process, save and print 8-bit, 16-bit and 32-bit images. It can read many image formats including TIFF, GIF, JPEG, BMP, DICOM, FITS and “raw”. It supports “stacks”, a series of images that share a single window. It is multithreaded, so time-consuming operations such as image file reading can be performed in parallel with other operations. As imageJ API’s [<http://rsbweb.nih.gov/ij/developer/api/>] are available, it is helpful to download & run it with own system (installed JVM).

**ImageJ API:** It has been used in counting no. of spikes

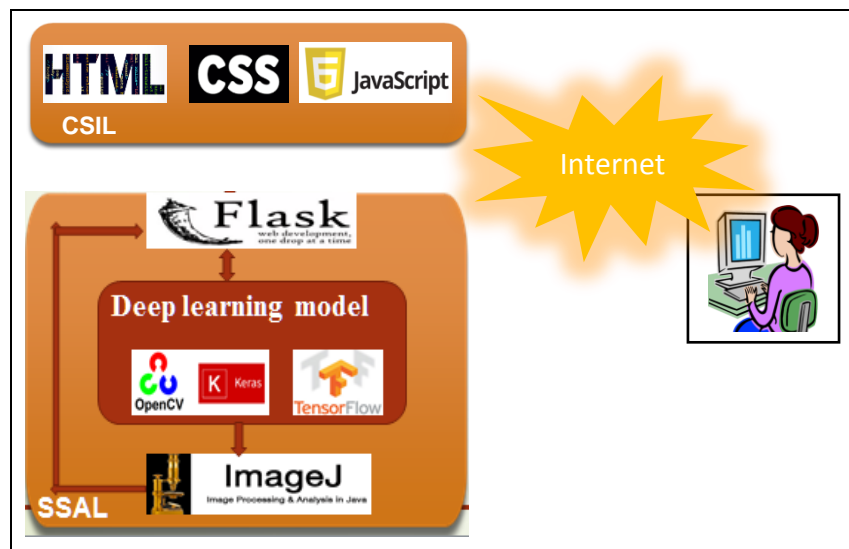
Class Analyzer

- Package: ij
- public class Analyzer extends java.lang.Object implements PlugInFilter, Measurements

This plugin implements ImageJ's Analyze/Measure and Analyze/Set Measurements commands which again implements flood-fill technique of image analysis.

### *Online software for spike identification and counting in wheat plant*

In this thesis, online software for identification and counting of wheat spikes has been developed using the following architecture (Fig 3.19).



**Fig 3.19** The architecture of the software

- **Client Side Interface Layer (CSIL):** It has been implemented using Hyper Text Markup language (HTML), Cascading Style Sheet (CSS) and JavaScript. User interface has been developed for capturing the information (*i.e.*, input images) from the user which will be used for spike identification and counting.
- **Server Side Application Layer (SSAL):** Application layer has been built using Flask web-development tool for taking the input from users, Deep learning module built with Tensorflow, Keras framework and Numpy, Scipy, Matplotlib,

OpenCV *etc.* libraries of python language for identification of spikes and ImageJ module to count the objects (or, spikes) on the output image of the deep learning module. After that, the output images with spike count output are again fed to the Flask API to communicate with the users.

### **Hyper Text Markup Language (HTML)**

HTML (Berners, 1990) is a platform independent simple text formatting language that is used to create hypertext documents. HTML tells the interpreter i.e. the browser, how the document has to be interpreted. The interpretive directions are given to the browser by a set of indicators called HTML elements, which helps to mark-up the document.

### **JavaScript**

JavaScript (Burns and Growney, 2001) was designed for extending the capabilities of Web browser. It also facilitates the web developers with an easy means of adding interactivity to their web sites. There are actually three essences of JavaScript: Core JavaScript, Client-Side JavaScript and Server-Side JavaScript. Core JavaScript considered as basic JavaScript language which includes the operators, control structures, objects *etc.* that makes JavaScript as a programming language. Client-Side JavaScript (CSJS) is used to provide access to the web-browser and web document objects via the Document Object Model (DOM). Another extension of Core JavaScript is Server-Side JavaScript (SSJS) which is embedded directly within HTML documents and runs on any SSJS-enabled Web server.

### **Cascading style sheets (CSS)**

The main purpose of Cascading Style Sheets (CSS) [<http://www.w3schools.com/css/>] is to allow the web authors in manipulating the appearance of web pages without affecting its HTML structure.

### **Flask**

Flask is a micro web framework which is written in Python [[https://en.wikipedia.org/wiki/Flask \(web framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))]. It was developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. It is classified as a micro-framework as it does not require particular tools or libraries. Flask supports extensions to add and update application features. Extensions involve

object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

## **PyCharm**

PyCharm is an integrated development environment (IDE) provides the facilities of computer programming, specifically for the Python language [<https://en.wikipedia.org/wiki/PyCharm>]. It is designed and developed by the Czech company JetBrains [<https://www.jetbrains.com/pycharm/>]. It facilitates of code analysis, graphical interface for debugging, integrated unit testing environment and also supports the web development facility. PyCharm is cross-platform, with Windows, macOS and Linux operating system. The Community Edition of Pycharm has been released under the Apache License [<https://www.eweek.com/development/jetbrains-strikes-python-developers-with-pycharm-1.0-ide>] and there is also Professional Edition of Pycharm with extra features and has been released under a proprietary license. PyCharm provides APIs to the developers to write their own plugins to extend the PyCharm features.

In this chapter, details of the materials and methodologies that have been used in developing image analysis based approach for measuring LFW in rice plant and spike identification and counting in wheat plant has been elaborated. Materials include image collection and dataset preparation for developing machine learning model whereas methodology section concerns with VIS and NIR image processing along with machine learning techniques (especially deep learning technique). The details about software tools and technologies that have used in developing macros and web bases software are also highlighted here. The next chapter (Chapter IV) contains the outcome and discussion of the proposed approach of Phenotyping parameter estimation.

## CHAPTER IV

### RESULTS AND DISCUSSION

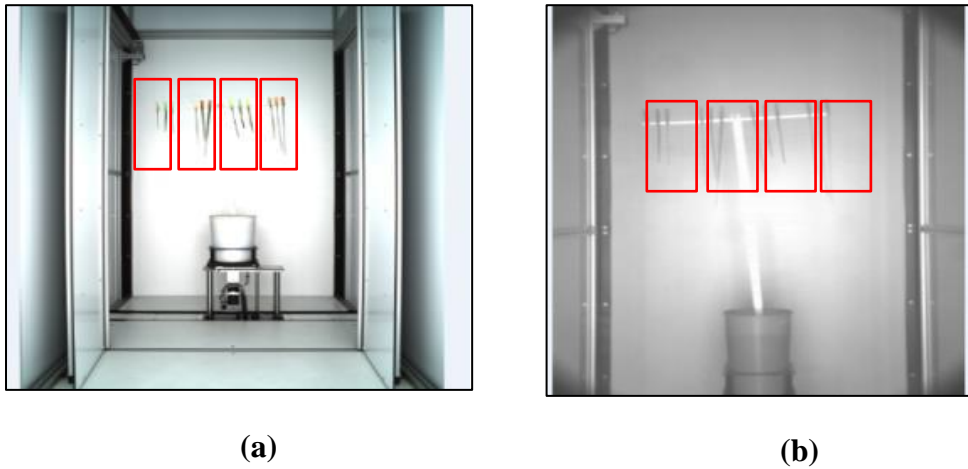
---

#### 4.1 Introduction

In this chapter, methodologies proposed in chapter III have been described with experimental datasets. Specifically, this chapter describes (i) Analysis of the proposed **VN\_LFW** approach of LFW estimation using VIS and NIR images and its comparative performance measurement with existing methods, (ii) Analysis of the proposed deep learning based approach **SpikeSegNet** and **LGspikeNet** for spike identification, (iii) counting approach and (iii) details description of the developed software.

#### 4.2 Analysis of VN\_LFW approach of LFW estimation

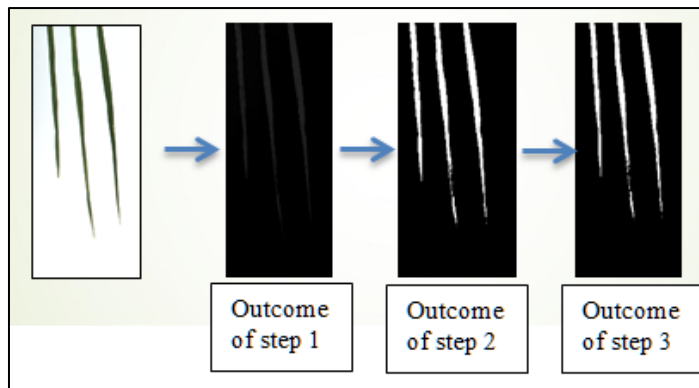
The proposed approach of LFW estimation is based on VIS and NIR imaging. Green Leaf Proportion (GLP) from VIS image and Mean Gray Intensity (NIR\_MGI) from NIR image have been used to develop Artificial Neural Network (ANN) model to estimate LFW in rice plant. In this study, the experimental dataset of 104 set of leaves [26 images and each with 4 set of leaves (Fig 4.1)] have been considered. Ground truth of LFW has been measured by using weighting machine. LFW values corresponding to each set are given in Annexure 1. The flow diagram of the proposed approach has been given in section 3.2.2 of chapter III.



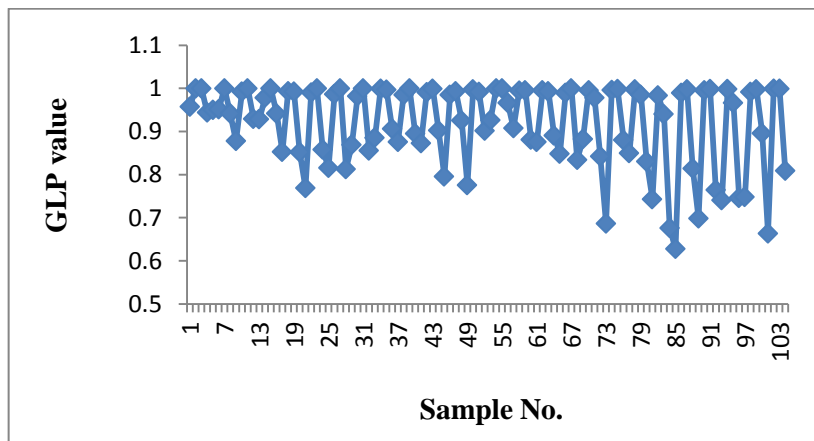
**Fig 4.1** Visual (a) and NIR (b) image and red boxes indicates the set of images

### ***Green Leaf Proportion (GLP) measurement from VIS image***

For GLP measurement from VIS image, the algorithm mentioned in the previous chapter of section 3.2.2 has been used and implemented in MATLAB software of version 2015b, MathWork. The macro (GLP.m) is given in Appendix I. GLP refers the ratio of green leaf area with respect to the total leaf area. Outcome of each step of the proposed algorithm is given in Fig 4.2. GLP values corresponding to 104 set of leaves are graphically represented in Fig 4.3.



**Fig 4.2** Outcome of each step of GLP measurement from VIS image

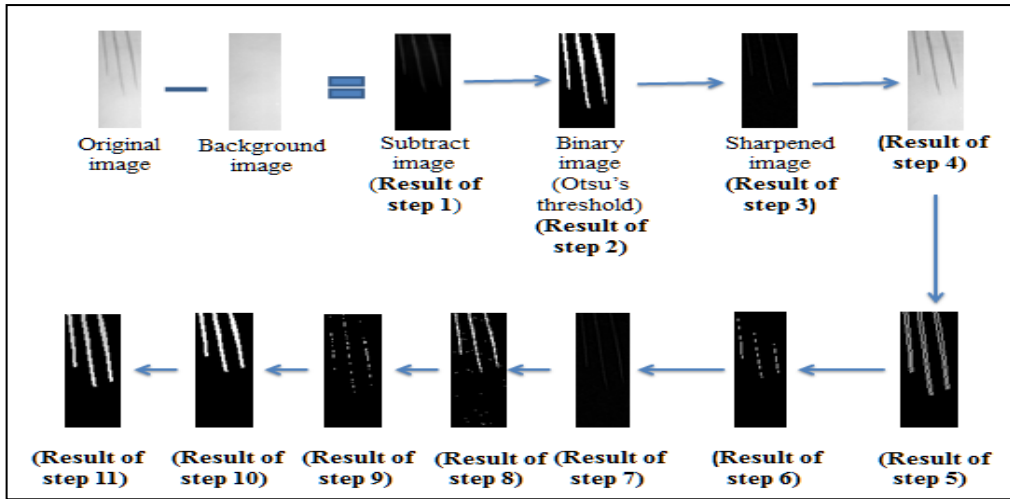


**Fig 4.3** GLP values corresponding to 104 set of leaves

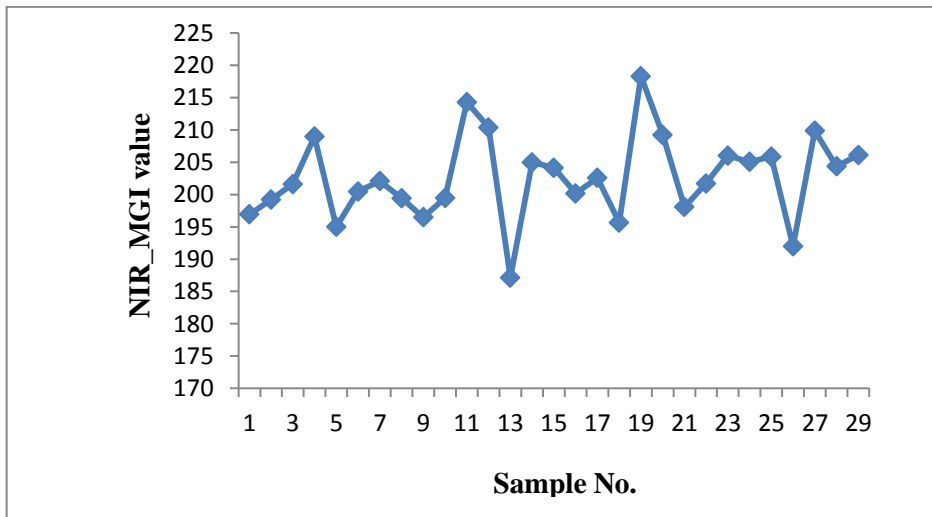
### ***Mean Gray Intensity (NIR\_MGI) measurement from NIR image***

For NIR\_MGI measurement from NIR image, the algorithm mentioned in the previous chapter of section 3.2.2 has been used and implemented in MATLAB software of version 2015b, MathWork [Fig 4.4 (a), (b)]. The macro (NIR\_MGI.m) is given in Appendix I. The outcome of each step is given in Fig 4.5. NIR\_MGI values corresponding to 104 set of leaves are given in Fig 4.6.





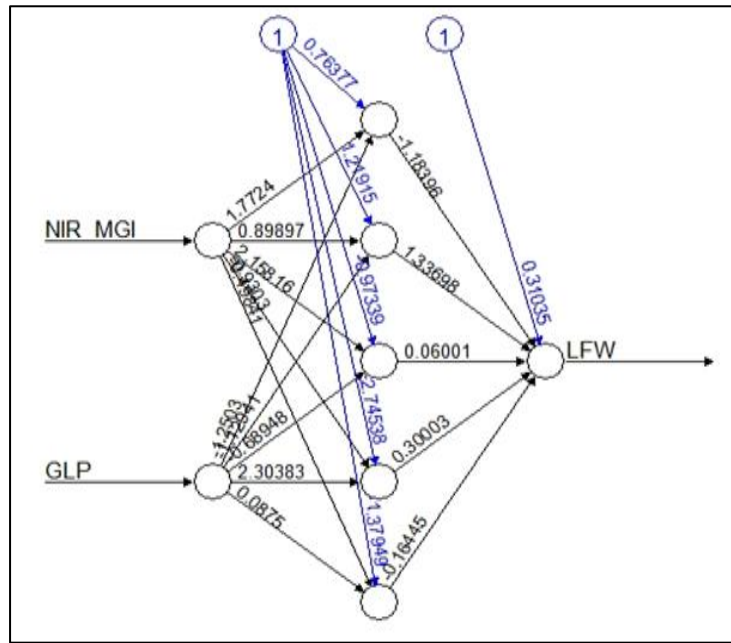
**Fig 4.5** Outcome of each step of gray value extraction from NIR image



**Fig 4.6** NIR\_MGI values corresponding to 104 set of leaves

#### 4.2.1 Model development and performance evaluation of VN\_LFW approach

For model development, the dataset (104 samples) has been divided into two parts randomly: 89 (*i.e.*, 85%) for training and 15 (*i.e.*, 15%) for testing. ANN model has been developed by using two independent parameters (GLP from VIS image and NIR\_MGI from NIR image) and one dependent variable (ground truth LFW). Distinctive combinations of hidden layer and hidden nodes in each layer have been attempted as there are no settled hypotheses accessible for the choice of hidden layer and hidden node. Out of which one hidden layers with 5 hidden nodes are performing superior than other combinations. “*Neuralnet*” package of R software (Teodoro, 2015) has been employed for development of the model. The architecture of the best fitted ANN model is given in Fig 4.7.



**Fig 4.7** Fitted ANN architecture

The proposed ANN approach has been compared with the conventional approaches (based on linear function of projected shoot area) as well as linear regression approach based on GLP from VIS image and NIR\_MGI from NIR image. Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are measured to compare the performances of the approaches both in training and testing dataset and the results are given in Table 4.1 and Table 4.2 respectively. The predicted LFW after applying ANN and regression technique in training and test dataset has been given in Annexure I.

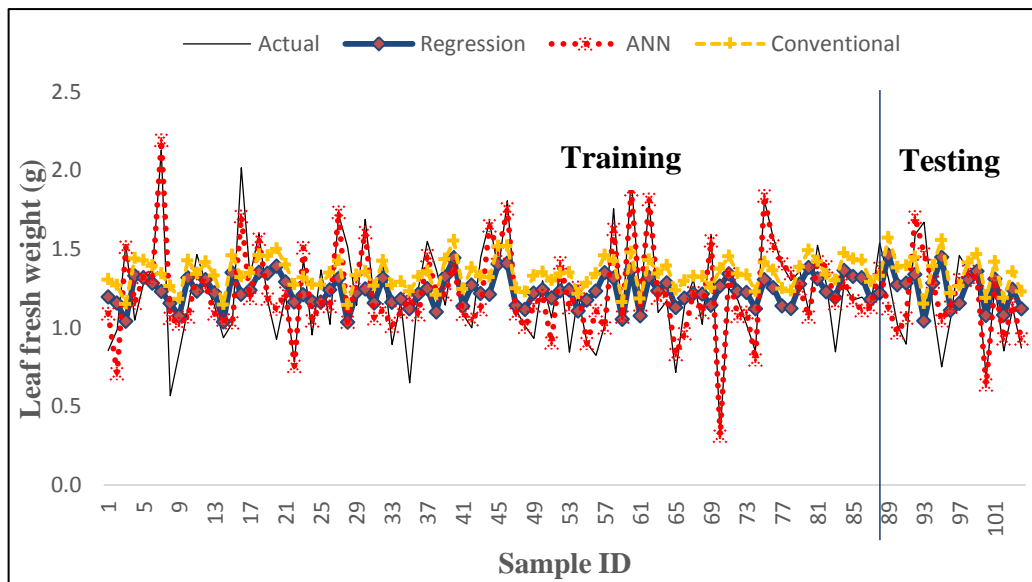
**Table 4.1** Comparison of proposed ANN based approach, regression and conventional approach in training dataset.

Indices of Prediction accuracy	Training		
	Conventional approach based on linear function of projected shoot area	Regression approach based on GLP and NIR_MGI	ANN approach based on GLP and NIR_MGI
<b>RMSE</b>	0.33	0.31	0.15
<b>MAPE</b>	27.66	23.44	9.55

**Table 4.2** Comparison of proposed ANN based approach, regression and conventional approach in testing dataset.

Indices of Prediction accuracy	Testing		
	Conventional approach based on linear function of projected shoot area	Regression approach based on GLP and NIR_MGI	ANN approach based on GLP and NIR_MGI
<b>RMSE</b>	0.36	0.31	0.13
<b>MAPE</b>	32.06	24.97	9.65

The above tables reflect that the proposed approach has outperformed than the other approaches in both training and testing dataset. The graphical plot of actual fresh weight of leaf versus fresh weigh predicted with different approaches (ANN, regression and conventional) are plotted in **Fig 4.8**. This also showed that the hypothesized approach with ANN modelling is superior as compared to other approaches.



**Fig 4.8** Line plots between actual LFW and modelled LFW (Regression, ANN and conventional approach) for the considered 104 samples

### 4.3 Analysis of the proposed approach of spike identification and counting in wheat plant

The proposed approach involves identification (section 4.3.1) and counting of spikes (section 4.3.2) from the digital images of wheat plant is discussed as follows:

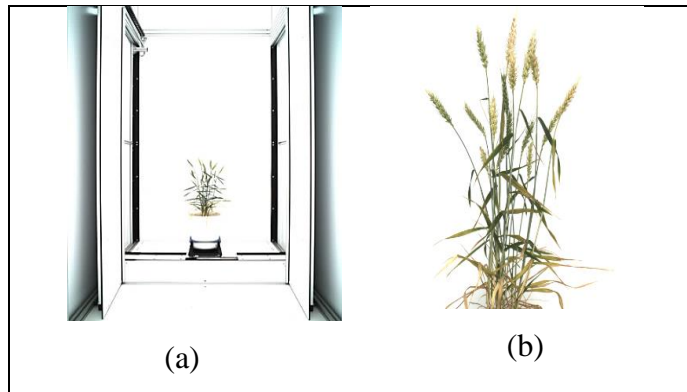
#### 4.3.1 Deep learning based approach for spike identification

Spike identification through image analysis is a pixel wise segmentation problem of object (here, spikes). Convolutional encoder-decoder deep learning technique based model plays a promising role in this aspect [Ronneberger *et al.* (2015); Szegedy *et al.* (2015); Mohanty *et al.* (2016); Jaswal *et al.* (2019)]. In this study, two deep learning models have been developed for spike identification namely **SpikeSegNet** (Spike Segmentation Network) and **LGspikeNet** (Local patch extraction and Global mask refinement spike detection Network) based on convolutional encoder-decoder deep learning technique as discussed in the previous chapter. Details of these two network architecture, dataset preparation for developing the network and performance evaluation are discussed below:

##### 4.3.1.1 SpikeSegNet - Spike Segmentation Network:

###### *Dataset preparation for developing SpikeSegNet*

Original size of the image is 6576 x 4384 pixels which consist of not only the plant regions but also the chamber used in imaging Fig 4.9a. So, the images are cropped to get only the region of interest (plant regions of size 1656\*1356) from the whole image as shown in Fig 4.9b.

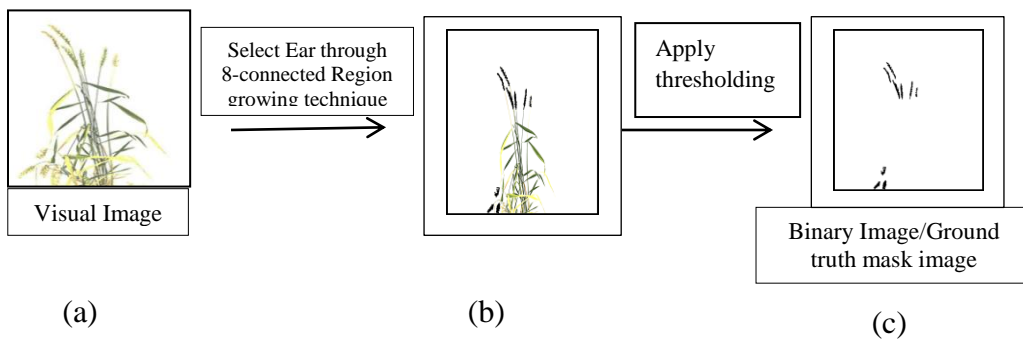


**Fig 4.9** a) Original image of plant with imaging chamber, b) Cropped visual image

For developing the proposed network, cropped images (of size 1656\*1356) are resized into 256\*256 to reduce the network complexity. Complexity of the network is proportional to the size of the image to be processed (Ronneberger *et al.*, 2015). For training the network, visual images (RGB image) and its corresponding ground truth images (usually mask image) with class label (*i.e.*, spike regions of the plant) has been prepared. Mask images consists of spike portions only corresponding to the visual image and are considered as part of network learning architecture for efficient identification of the objects (*i.e.*, spike regions). For masking the image following steps have been involved:

*Step 1:* “wand tool” of photoshop software has been used to create a selection by tracing objects of uniform color which implements 8-connected region growing technique (Fig 4.10b).

*Step 2:* Thresholding the resultant image of *step 1* (Fig 4.10c).

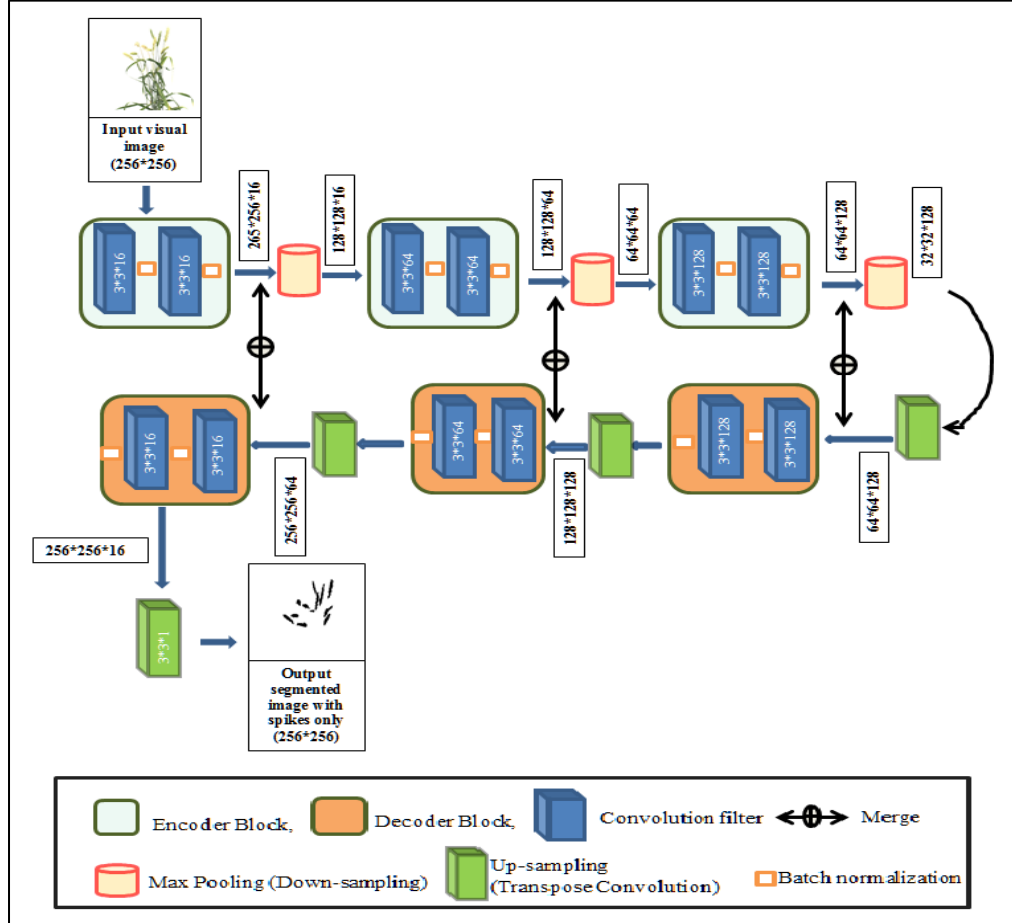


**Fig 4.10** a) Visual image, b) Selection of spike regions of the image, c) Threshold binary image or mask image

### ***Development of SpikeSegNet network***

In this experiment, total 600 images of 200 plants from 3 side directions were recorded. Image dataset of randomly selected 85% of the total plant (*i.e.*, 510 images of 170 plants) have been used in developing the model to identify or recognize spike regions on the digital image (binary masks) of the plant. The developed model has been tested on the randomly selected 15% of the dataset (*i.e.*, 90 images of 30 plants). The network is developed for pixel wise segmentation of objects (or spikes) from the whole plant of wheat. The proposed network has been trained by using visual images and its corresponding masks images containing class label (*i.e.*, spike regions of the plant). In this study, the proposed network consists of 3 encoder

blocks and corresponding hierarchy of 3 decoder blocks as shown in Fig 4.11. The numbers of encoder and decoder blocks are estimated empirically to yield the best results with optimum performance and by taking inspiration from UNet architecture (Ronneberger *et al.*, 2015) popularly used in biomedical image segmentation.



**Fig 4.11** Details architecture of the proposed SpikeSegNet deep-network

Each **encoder** block consists of convolution layer to produce a set of feature maps with rectified linear non-linearity (ReLU) activation function (Agostinelli *et al.*, 2014). Then these feature maps are batch normalized (Ioffe & Szegedy, 2015) to improve the performance and stability of the network and followed by max-pooling with 2\*2 window with stride 2 (filter/kernel moves across and down a 2 pixel) has been used for sub-sampling or down-sampling the features by factor of 2. Each encoder block is repeated with varying filter depth of 16, 64, and 128 to encode the features. Square filters have been used as it is popularly used in various state-of-art methods (Simonyan & Zisserman, 2014). Details of each encoder block (*i.e.*, input to the each encoder block, number of convolution filter used with their sizes, output of

each encoder block, input and output to the corresponding max-pool) are given in Table 4.3.

**Table 4.3** Details of each encoder block and corresponding max-pool

Encoder Block #	Input to encoder block	Convolution filter size	Number of convolution filter	Output of encoder block	Input to max-pool	Output to max-pool
Block 1	256*256 (original image)	3*3	16	256*256*16	256*256*16	128*128*16
Block 2	128*128*16	3*3	64	128*128*64	128*128*64	64*64*64
Block 3	64*64*64	3*3	128	64*64*128	64*64*128	32*32*128

For **decoding**, convolutional 2D transpose (or, de-convolution) has been used to up-sample the feature maps by a factor of 2 as opposite to the max-pool. The spatial information from the encoder has been forwarded to the decoder along with incoming up-sampled features which helps in generating more location specific masks. Each decoder block consists of convolution layer with “ReLU” activation function to produce features followed by batch normalization has been done to the each produced feature. Each decoder blocks are repeated for varying channel depths of 128, 64, and 16 to decode the features. Details of each decoder block (*i.e.*, input to the each decoder block, number of convolution filter used with their sizes, output of each decoder block, input and output to the corresponding transpose convolutional layer) are given in Table 4.4.

**Table 4.4** Details of each decoder block and corresponding transpose convolutional layer

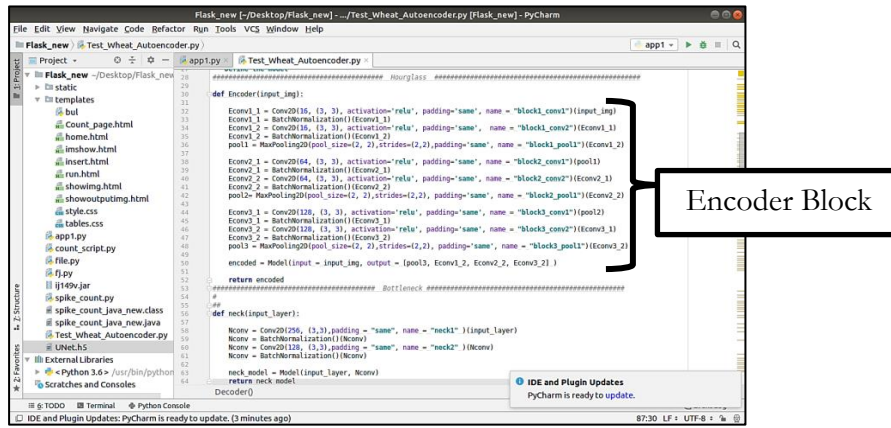
Decoder Block #	Input to transpose convolution	Output of transpose convolution	Input to decoder block	Convolution filter size	Number of convolution filter	Output of decoder block
Block 1	32*32*128	64*64*128	64*64*128	3*3	128	64*64*128
Block 2	64*64*128	128*128*64	128*128*64	3*3	64	128*128*64
Block 3	128*128*64	256*256*64	256*256*64	3*3	16	256*256*16

Output of the final decoder has been fed into 3\*3\*1 convolution layer with “softmax” activation function (Dunne & Campbell, 1997) to classify the object (*i.e.*, spike). “Adam” optimizer (Kingma & Ba, 2014) was used with a learning rate of 0.0005 to update the weights. “Binary cross-entropy” (Hagenauer *et al.*, 1996) loss function has been used in this study to predict binary class label (*i.e.*, spikes and non-spikes). It is the most commonly used loss function in image segmentation to

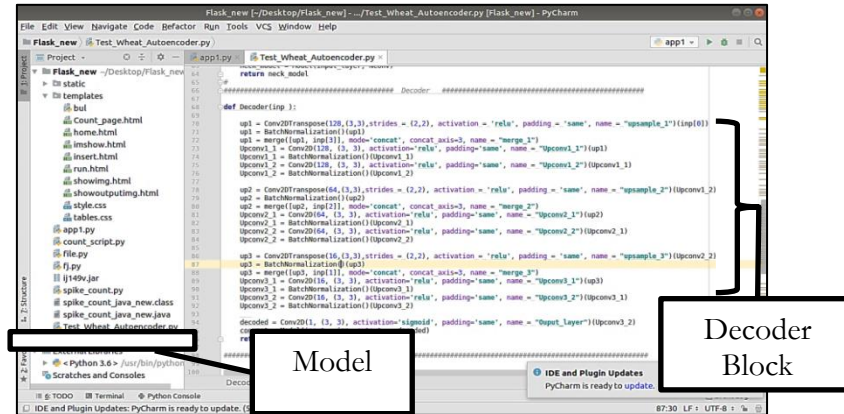
compute pixel-wise cross entropy. It examines each pixel individually and compares its binary class predictions (either 0/1; in this context, spikes region or not) to the ground truth (or, segmented ground truth consisting of spike regions only). Cross entropy loss evaluates the class predictions for each pixel individually and averages it over all the pixels. Therefore, each pixel contributes uniformly to the overall objective loss function. Suppose there is wheat plant image  $I$  of size  $p \times q$ , corresponding ground truth mask is  $I^{grtr}$  and the mask image predicted by the developed network is  $I^{pred}$ . Hence, binary cross entropy loss  $Loss\_Binary\_Cross\_Entropy(I^{pred}, I^{grtr})$  can be defined as follows:

$$Loss\_Binary\_Cross\_Entropy(I^{pred}, I^{grtr}) = -\frac{1}{p * q} \sum_{l=1}^q \sum_{k=1}^p [I^{grtr}(p, q) * \log(I^{pred}(p, q)) + (1 - I^{grtr}(p, q)) * \log(1 - (I^{pred}(p, q)))] \quad (vi)$$

The network has been developed and implemented by using python libraries (such as, Keras, TensorFlow, Matplotlib *etc.*) in PyCharm editor (**Fig 4.12**).



(a)

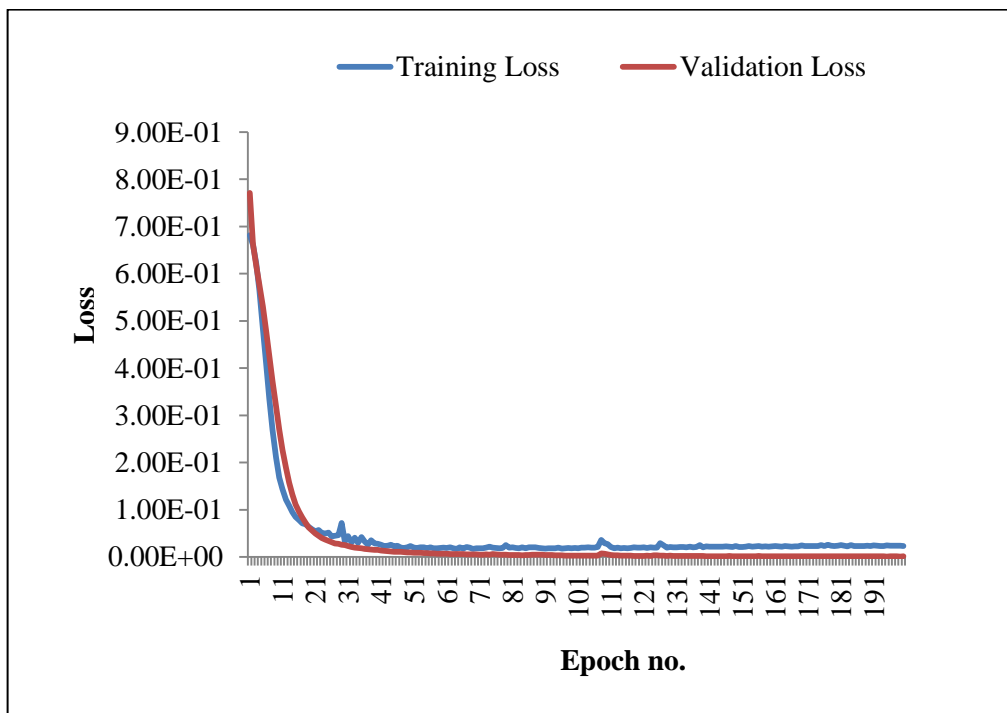


(b)

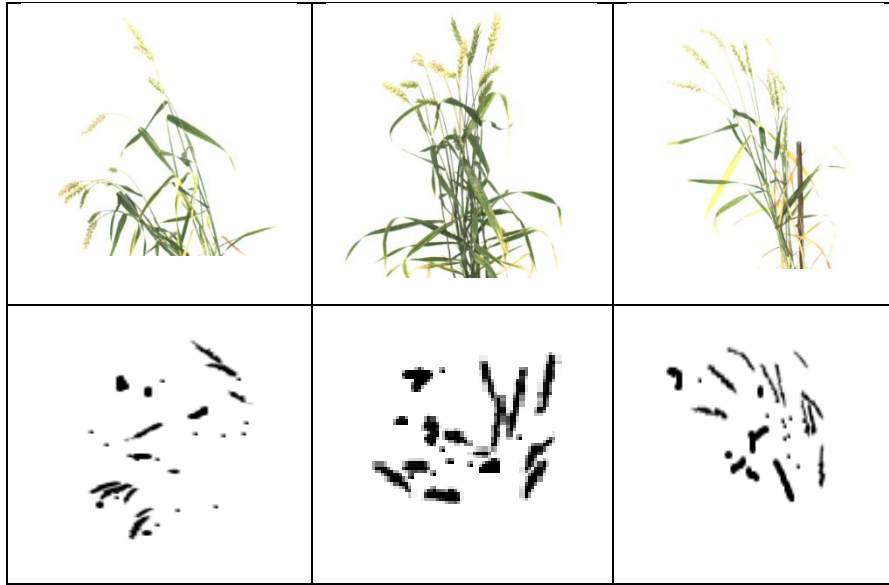
**Fig 4.12 (a), (b) Training of encoder-decoder network in PyCharm**

### ***Performance evaluation of SpikeSegNet***

Training of the network has been done for 200 epochs and at each epoch training and validation loss has decreased and plateaued around 160 epochs (Fig 4.13). The performance analysis of the proposed segmentation network for identification or detection of spikes has been tested on the randomly selected validation dataset (*i.e.*, 90 images from 3 side-directions of 30 plants). For this purpose, the dataset has been fed to the developed model of the trained network and the resulting binary mask images are compared with the ground truth images and thereafter, segmentation performance is measured by different performance parameters. The different evaluation parameters like  $E_1$ ,  $E_2$ , Jaccard Index (JI), Accuracy, Precision, Recall, and F-measure are computed and average value of these parameters for the 90 images of 30 plants has been shown in Table 4.5. Details about statistical evaluation parameters are given in section 3.3.3.1 of chapter III. Predicted output masks of some of the sample test images are shown in Fig 4.14.



**Fig 4.13** Training and validation loss per epoch



**Fig 4.14** Predicted output masks of some of the sample test images

**Table 4.5** Metrics for identification of spikes

<b>E1</b>	<b>E2</b>	<b>Jaccard Index (JI)</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F-measure</b>
0.03061	0.13356	0.94348	0.94669	0.94561	0.94786	0.94887

The average value of the segmentation error  $E_1$  and  $E_2$  are 0.03061 and 0.13356 respectively (Table 4.5). As the performance has been calculated at pixel level, the error seems to be statistically significant. Accuracy of the developed model is near to 95% and spikes are detected with an average precision and recall of 94.56% and 94.78%, respectively. The precision value reflects that 94.56% of the detected pixels are actually spikes, whereas recall value reflects that 94.78% of actual spikes pixels were detected among the ground truth spike pixels using the developed network. Average  $F_1$  score reveals that the proposed network is 94.88% robust in identifying or detecting spikes from whole plant images.

#### **4.3.1.2 LGspikeNet - Local patch extraction and Global mask refinement spike detection Network:**

While carrying out the experiment, it was observed that **SpikeSegNet** resulted in some blurriness and irregular shape of the object (Fig 4.14). This might have arisen as the network has been trained end-to-end by resizing the original image

into  $256 \times 256$ ; it loses the contextual as well as spatial information which is very essential to learn features for the segmentation network. To improve upon the same a new deep network **LGspikeNet** (Local patch extraction and Global refinement spike detection Network) has been proposed for spike identification at patch level. Patches are nothing but the small size parts of an image as shown in Fig 4.16. **LGspikeNet** is combination of two especial features as follows:

- 1) Local Patch Extraction (LPspikeNet)
- 2) Global Mask Refinement (GMspikeNet)

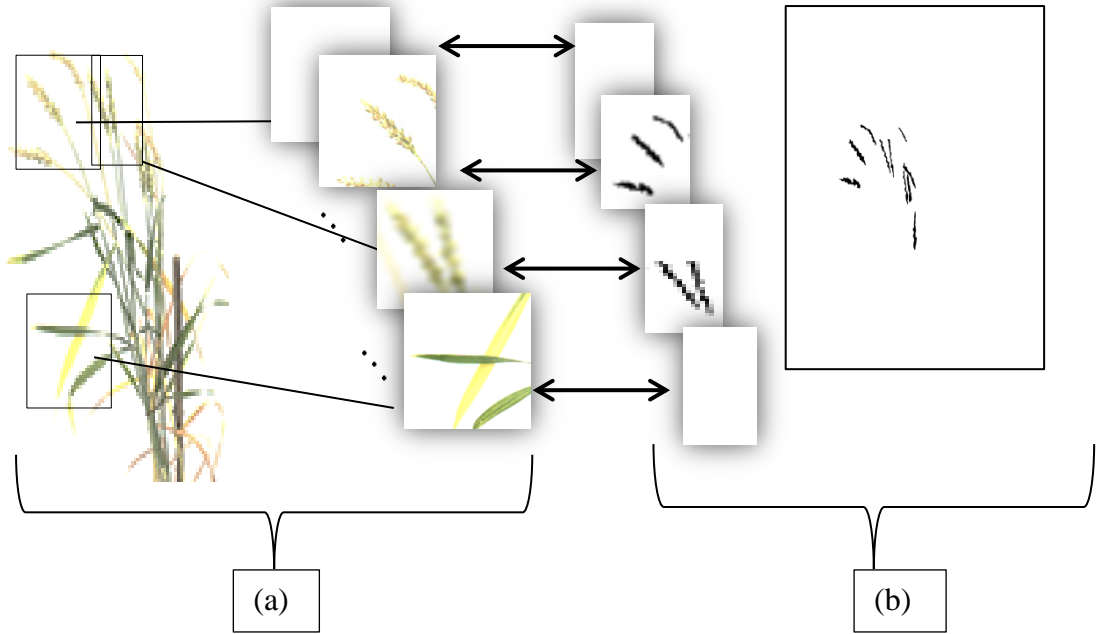
In LPspikeNet, the contextual and spatial features are learned at local patch level. The output of the network is segmented mask image which are then refined at global level using GMspikeNet. The details of the dataset preparation for patch level feature learning and the network architecture are given below:

***Dataset preparation for developing LGspikeNet:***

For dataset preparation same procedure has been followed as mentioned in the sub-section of 4.3.1 for **SpikeSegNet**. After that, visual images (of size  $1656 \times 1356$ ) as well as ground truth mask images (of size  $1656 \times 1356$ ) are divided into 100 pixel overlapping patches of size  $256 \times 256$ . So, from one image 180 patches will be generated. In this process from one image 180 patches will be generated which in turn will act as images.

**Fig 4.15** Original image (of size  $1656 \times 1356$ ) is divided into 100 pixel overlapping patches where red box is of size  $256 \times 256$

	0	1	2	3	100	101	...	256	...	356		...	1400	...	1656
0															
1															
2															
3															
.															
.															
.															
1356															



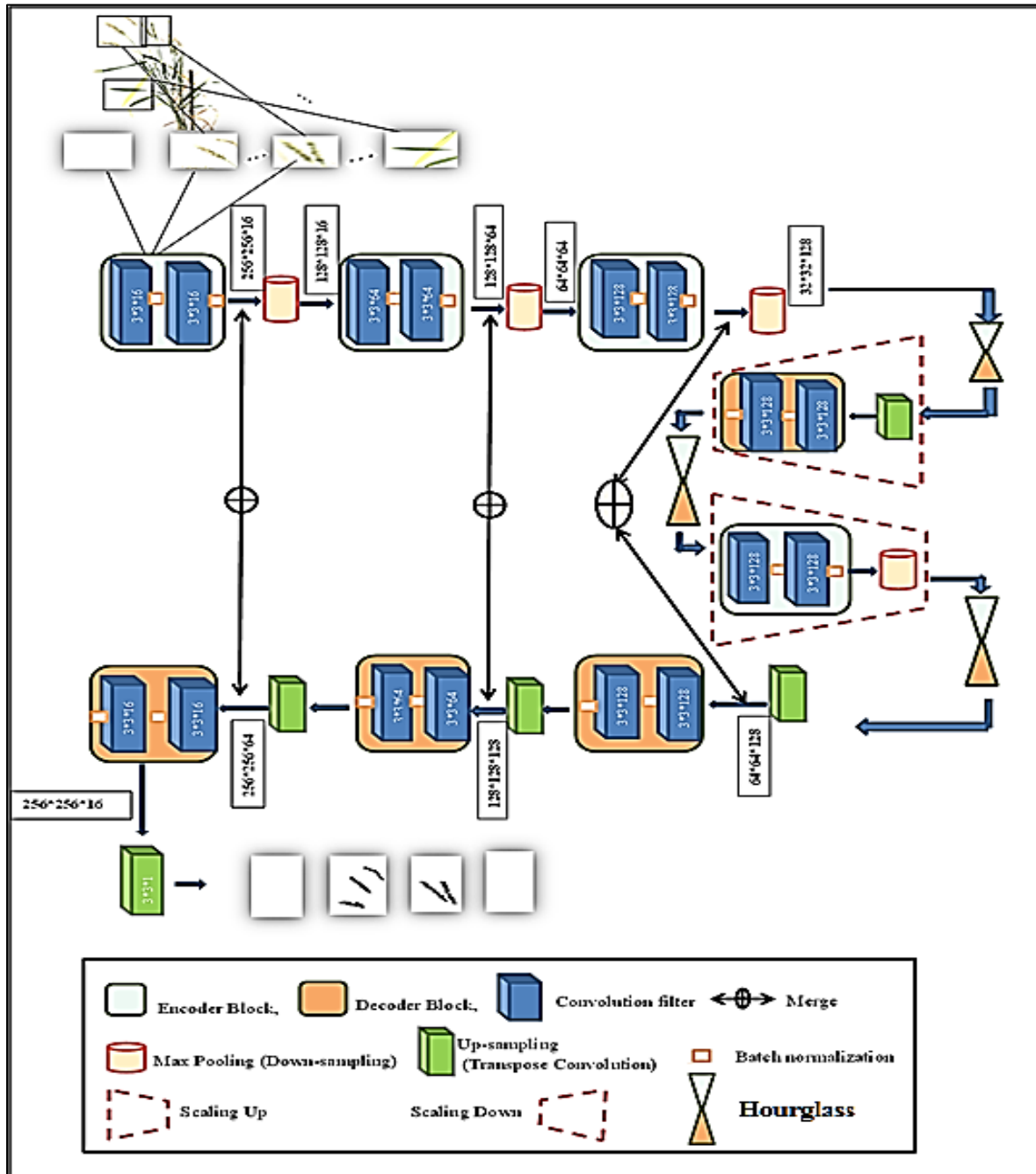
**Fig 4.16** (a) Patches of visual image and (b) corresponding mask image

### ***Architecture of LGspikeNet:***

#### ***1. Architecture of Local Patch Extraction Network (LPspikeNet):***

The network architecture consists of Encoder, Decoder and Hourglass as bottleneck of the Encoder and Decoder. The architecture of the network has been built by taking inspiration from UNet (Ronneberger *et al.*, 2015). Encoder takes input patch images to give feature map representation that holds the contextual and spatial information. Decoder takes the information as input and produce corresponding segmentation masks as output. Hourglass is introduced in the bottleneck to compress the feature map representation for better segmentation results. Additionally, skip connections [Ronneberger *et al.* (2015); Jaswal *et al.* (2019)] are formed between the encoder and the decoder. The skip connection helps in transferring the spatial information across the network for better localization of the segmentation masks. Through the skip connections, corresponding feature maps from the encoder before down sampling (or, max-pooling) are concatenated with the corresponding feature maps of the decoder after up-sampling (or, transverse convolution). The architecture of the proposed **LPspikeNet** network consists of 3 encoder blocks, corresponding hierarchy of 3 decoder blocks and 3 hourglasses between encoder-decoder as bottleneck. By introducing hourglass in the bottleneck,

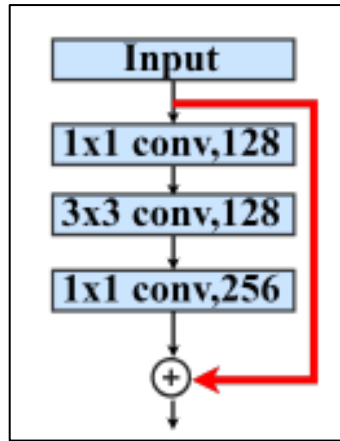
the segmentation network gives precise and contextually more confident segmentation mask (Ronneberger *et al.*, 2015). But, more than one hourglass in the network will increase the network depth as well as performance may fall due to over-fitting. The numbers of encoder blocks, decoder blocks and the hourglass are estimated empirically to yield the best results with optimum performance and by taking inspiration from UNet architecture. The architecture is given in Fig 4.17 and the details of encoder, decoder and hourglass are given in the following sections.



**Fig 4.17** Details architecture of the proposed **LPspikeNet** network

**Encoder (E):** The 1<sup>st</sup> encoder block takes the image patches as input and produces corresponding feature maps as output and forwarded it to the 2<sup>nd</sup> and 3<sup>rd</sup> encoder block for further feature extraction. Each encoder block consists of repeated sets of dual 3\*3 convolutions (with padding 1 and stride 1) succeeded by ReLU activation and Batch Normalization. Each block is succeeded by a max-pool operation (size 2 \* 2 and stride 2) and is repeated with filters having varied channel depths of 16; 64; 128. The design architecture is same as discussed in subsection of 4.3.1 and **Table 4.4**.

**Hourglass (H):** By introducing hourglass network as the bottleneck, it gives more confident segmentation result with lesser blurriness [Ronneberger *et al.* (2015)]. It's mainly due to the innate design of hourglass network which minimizes the feature map and captures the information by only concentrating on essential features. Multiple hourglass networks enhance the invariant features that are captured at various scale, viewpoint and occlusion very effectively to predict the segmentation mask of the image accurately [Jaswal *et al.* (2019)]. The output of the encoder network has been passed as input to the hourglass network. Along with local context, it captures relationship among different local patterns (*i.e.*, global context). The network consists of two parts: **Hourglass Encoder (Hg<sup>E</sup>)** and **Hourglass Decoder (Hg<sup>D</sup>)**. Each encoder and decoder in hourglass again contains **Residual Module/Block** to resolve the problem of over-fitting as well as gradient vanishing issues. In simple Encoder/Decoder network, after each max pool step the output of encoder block is concatenated with the corresponding decoder block. In hourglass network, instead of concatenating the layer of the encoder with that of the decoder, the layer is further convolved through residual block and then added element-wise to the corresponding layer of the decoder. The **Residual Module/Block** consists of a 1 \* 1 convolution of depth 128 followed by 3 \* 3 convolution of depth 128 and then 1 \* 1 convolution of depth 256 as shown in Fig 4.18. **Hg<sup>E</sup>** network receives the output from the encoder network **E** and contains four residual modules in sequential order and **Hg<sup>D</sup>** network contains long-term skip connections in order to preserve the spatial information. Similar to **Hg<sup>E</sup>**, it also contains four residual modules in sequential order. Input and output of each hourglass has been given in Table 4.6.



**Fig 4.18** Structure of single residual block

**Table 4.6** Input and output of each hourglass

Hourglass	Input	Output	After Scale Up	After Scale Down
Hourglass 1	32*32*128	32*32*128	64*64*128	—
Hourglass 2	64*64*128	64*64*128	—	32*32*128
Hourglass 3	32*32*128	32*32*128	—	—

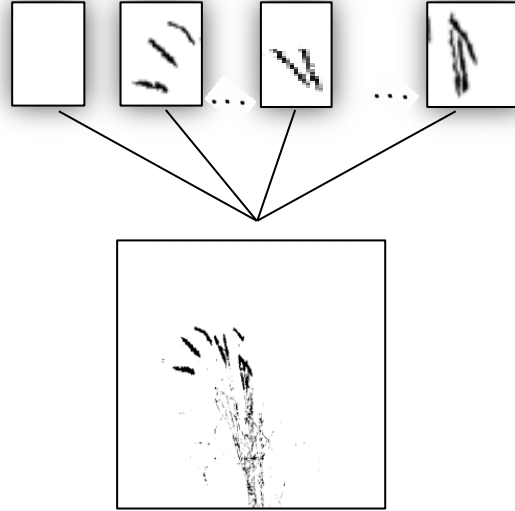
“—” indicates the corresponding operation has not been done

**Decoder (D):** In decoder, the output from the 3<sup>rd</sup> hourglass (32 \* 32 \* 128) is up-sampled using a 3\*3 transpose convolution with padding 1 and stride 1. Then the resulting feature map (of size 64 \* 64 \* 128) got concatenated with the corresponding encoder feature map. The concatenated feature map (of size 64 \* 64 \* 256) is then passed to two, 3 \* 3 convolution layers (padding 1 and stride 1) followed by ReLU activation and Batch Normalization and is repeated with filters having varied channel depths of 128, 64 and 16 as opposite to the encoder blocks. The design architecture is same as discussed in subsection of 4.3.1 and Table 4.4.

Output of the final decoder has been fed into 3\*3\*1 convolution layer with “softmax” activation function to classify the object (i.e., spike) at patch level. “Adam” optimizer was used with a learning rate of 0.0005 to update the weights. “Binary cross-entropy” was used as loss function to predict binary class label (i.e., spikes and nonspikes) at patch level. Sample outputs of **LPspikeNet** are already given in Fig 4.17.

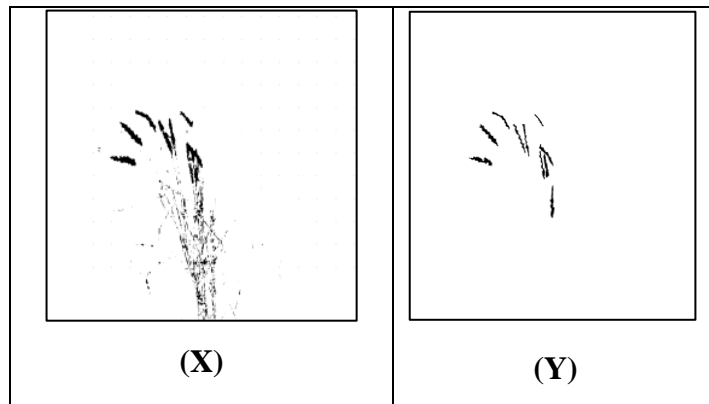
## B. Global Mask Refinement Network (GMspikeNet):

Architecture of **GMspikeNet** is same as **SpikeSegNet** as discussed in section 4.3.1 of this chapter. Output of **LPspikeNet** is predicted mask image of size  $256 \times 256$  of the visual image patches as shown in Fig 4.17. These mask image patches then merged to construct the predicted mask image (*mergeLPmask*) of original size ( $1656 \times 1356$ ) as represented in Fig 4.19.



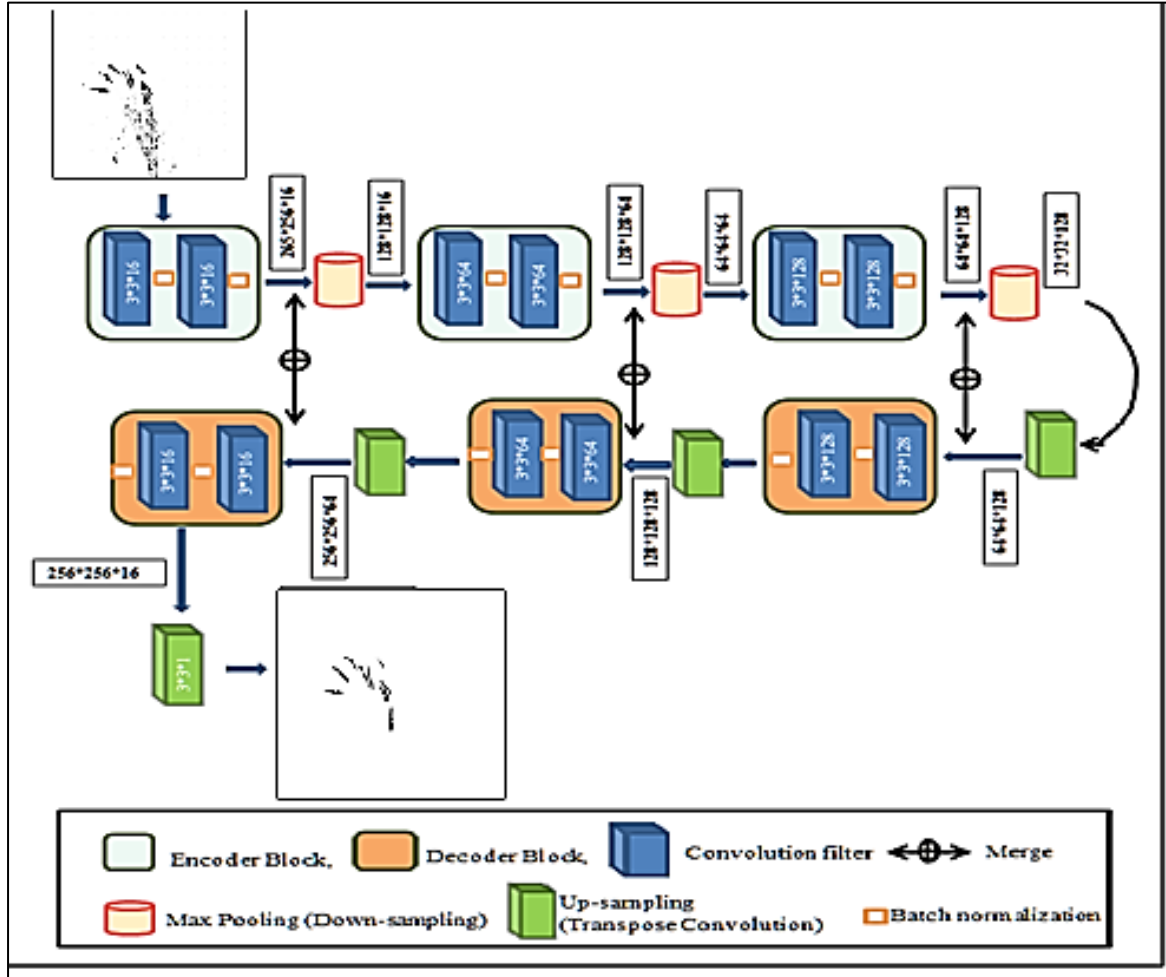
**Fig 4.19** Merging of mask image patches (*mergeLPmask*)

As shown in the above diagram, output of **LPspikeNet** contains blurriness. To resolve the problem **GMspikeNet** has been developed for refining the mask image (*mergeLPmask*). After that *mergeLPmask* (of size  $1656 \times 1356$ ) has been resized into  $256 \times 256$  and used as input/dependent variable (X) to produce actual ground truth mask image (Y) as output/dependent variable as shown in Fig 4.20.



**Fig 4.20** Independent (X) and dependent (Y) variable for training **GMspikeNet**

The proposed **GMspikeNet** network consists of 3 encoder blocks and corresponding hierarchy of 3 decoder blocks. The inner-structure and hyper-parameter of the encoder/decoder block is same as given in Table 4.3 and Table 4.4. Details architecture of **GMspikeNet** is given below (Fig 4.21).

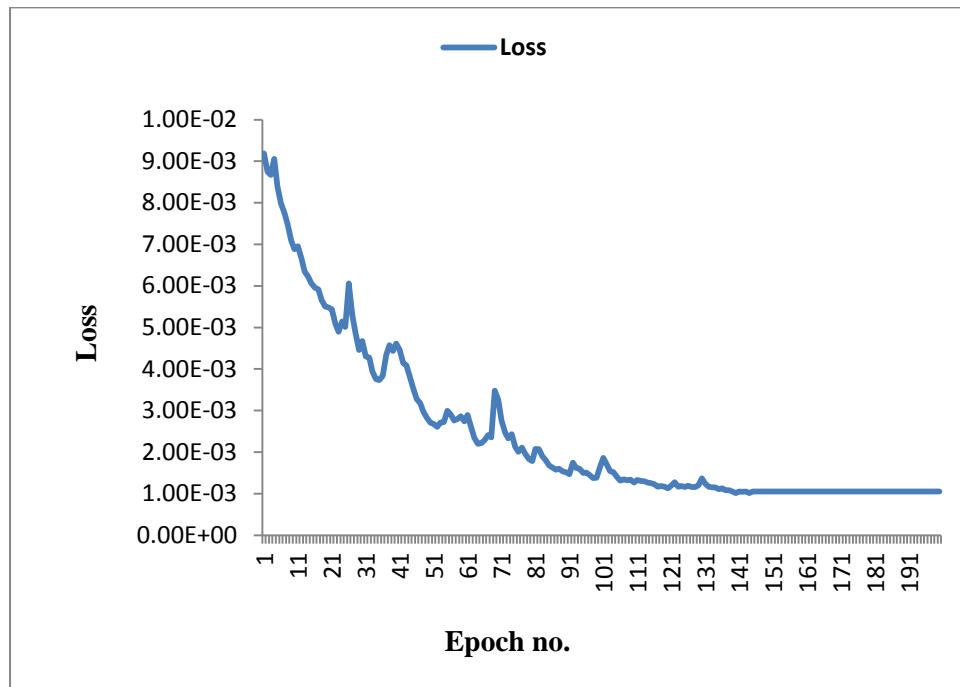


**Fig 4.21** Details architecture of the proposed **GMspikeNet** network

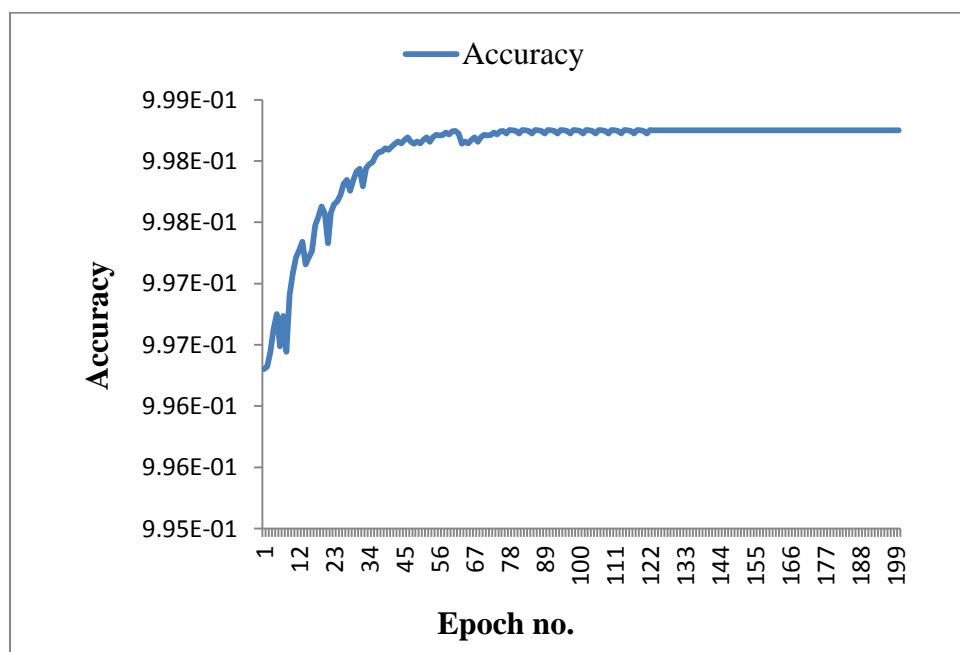
#### **Training and performance evaluation of LGspikeNet:**

For training **LGspikeNet**, 85% of the total plant images (*i.e.*, 510 images out of 600 images) have been used. First part of the network *i.e.*, **LPspikeNet** has been trained by extracting patches from the training image dataset as mentioned in **section 4.3.2**. As original image (of size 1656\*1356) and its corresponding ground truth mask image (of size 1656\*1356) is divided into 100 pixel overlapping patches, so, the training dataset contains 91800 patches (180 patches of 510 images each). Training of the network has been done for 200 epochs and the loss has been computed at each epoch. At every epoch it was decreased and after 145 epochs it was plateaued (Fig

4.22). Average accuracy has also been computed for each epoch (Fig 4.23), and around 118 epochs it was plateaued and shown high accuracy in identifying the mask image at patch level.



**Fig 4.22** Training loss per epoch



**Fig 4.23** Training accuracy per epoch

Sample of the predicted output mask image patches are already shown in Fig 4.17. Then the corresponding mask image patches are merged to construct original size of the mask image of the training dataset (*i.e.*, 510 mask image of size 1656\*1356). As discussed in the previous section, output of **LPspikeNet** contains blurriness. For refining these blurriness second part of the **LGspikeNet** network *i.e.*, **GMspikeNet** has been trained. The training dataset contains 510 output images of **LPspikeNet** as independent variable (X) and its corresponding ground truth mask image as dependent variable (Y) (Fig 4.20). Predicted output masks of some of the sample test images after applying **GMspikeNet** are shown in Fig 4.24.



**Fig 4.24** Predicted output masks of some of the sample test images

The performance analysis of the proposed segmentation network **LGspikeNet** for identification or detection of spikes was tested on the same validation dataset (*i.e.*, 90 images from 3 side-directions of 30 plants) as done in case of **SpikeSegNet** network. For this purpose, the dataset was fed to the developed model of the trained network and the resulting binary mask image was compared with the ground truth image and thereafter, segmentation performance was measured by different performance parameters. The different evaluation parameters like  $E_1$ ,  $E_2$ , Jaccard Index (JI), Accuracy, Precision, Recall, and F-measure are computed and average value of these parameters for the 90 images of 30 plants has been shown in Table 4.7.

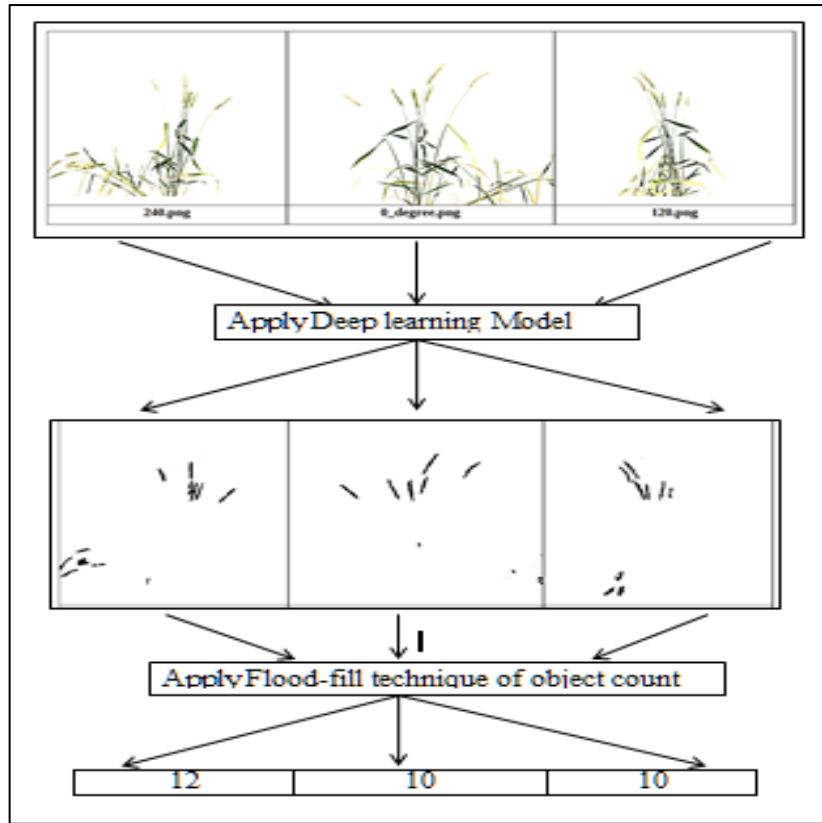
**Table 4.7** Metrics for identification of spikes

<b>E1</b>	<b>E2</b>	<b>Jaccard Index (JI)</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F-measure</b>
0.000608	0.033561	0.999387	0.999693	0.999516	0.999871	0.999693

Accuracy of the developed model is near to 100% and spikes are detected with an average precision and recall of 99.95% and 99.98%, respectively. The precision value reflects that 99.95% of the detected pixels are actually spikes, whereas recall value reflects that 99.98% of actual spikes pixels were detected among the ground truth spike pixels using the developed network. Average  $F_1$  score reveals that the proposed network is 99.96% robust in identifying or detecting spikes from whole plant images.

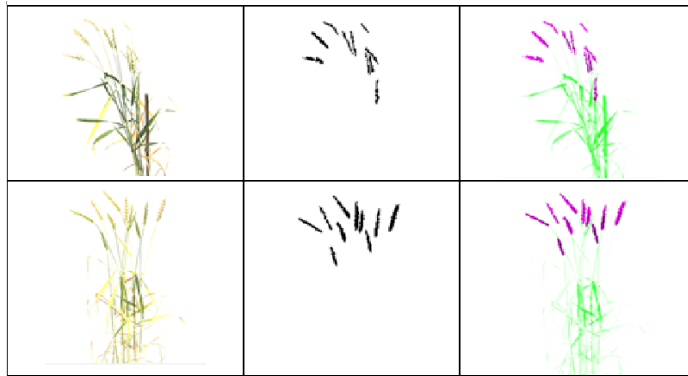
#### 4.3.2 Analysis of spike count approach

For counting number of spikes per plant, VIS images from three directions ( $0^0$ ,  $120^0$ ,  $240^0$ ) of the plant have been used because image from one direction cannot cover all the spikes. Then the three direction VIS images corresponding to one plant are passed as input to the developed **LGspikeNet** model. **LGspikeNet** network model has been chosen for spike detection as it out-performs than **SpikSegNet**. Output of the model is nothing but the predicted mask image/ binary image. After that, “*analyse particles*” function of imageJ has been applied on the output images to count number of spikes/ears (Schneider *et al.*, 2012). Flow diagram of approach is presented in Fig 4.25. Performance of the developed model has been tested on the same validation dataset (15% of the dataset *i.e.*, 90 images of 30 plants). It has been found that, the maximum spike count obtained from the images of three directions ( $0^0$ ,  $120^0$ ,  $240^0$ ) of the single plant is very closely associated with the ground truth spike count. Hence, the image with maximum spike count has been used to compare and evaluate the performance of the proposed approach for spike counting.



**Fig 4.25** Flow diagram of counting spike number in a single plant

In order to validate the counting approach, resultant output mask images were superimposed over the original image (RGB image) and TP, FP, FN, *etc.* was computed. Some of the test images, their output mask image and corresponding superimposed images are as shown in **Fig 4.26**.



**Fig 4.26** (a) original image (RGB image), (b) Predicted output masks image, (c) superimposed image

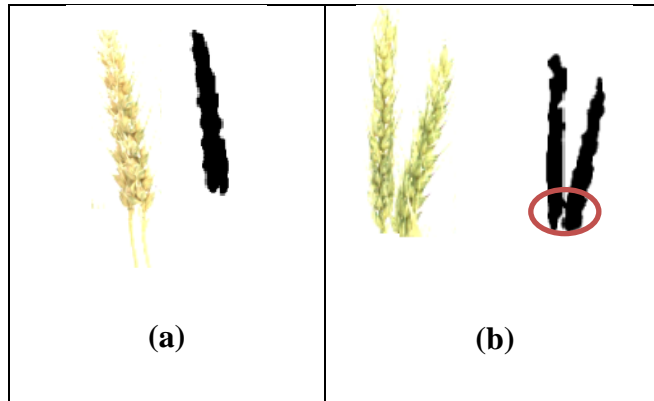
The precision, accuracy and  $F_1$  score corresponding to the 30 plants are represented in **Table 4.8**. Average precision, accuracy and  $F_1$  score was 99%, 94% and 96%, respectively.

**Table 4.8** Outcome of the proposed approach of spike counting on the test images of 30 plants

Image no.	Ground Truth	Predicted using Model	TP	FP	FN	Precision	Accuracy	F <sub>1</sub> score
1	10	9	9	0	1	1.00	0.90	0.95
2	8	7	7	1	1	0.88	0.78	0.88
3	10	9	8	0	1	1.00	0.89	0.94
4	11	10	10	0	1	1.00	0.91	0.95
5	10	10	10	0	0	1.00	1.00	1.00
6	9	9	8	1	0	0.89	0.89	0.94
7	10	9	8	1	1	0.89	0.80	0.89
8	6	6	6	0	0	1.00	1.00	1.00
9	12	11	10	0	1	1.00	0.91	0.95
10	12	12	11	0	1	1.00	0.92	0.96
11	13	12	10	0	1	1.00	0.91	0.95
12	11	10	9	0	1	1.00	0.90	0.95
13	6	6	6	0	0	1.00	1.00	1.00
14	8	8	7	1	0	0.88	0.88	0.93
15	16	15	13	2	1	0.87	0.81	0.90
16	2	2	2	0	0	1.00	1.00	1.00
17	10	10	10	0	0	1.00	1.00	1.00
18	1	1	1	0	0	1.00	1.00	1.00
19	11	10	10	0	0	1.00	1.00	1.00
20	7	7	7	0	0	1.00	1.00	1.00
21	8	7	7	0	1	1.00	0.88	0.93
22	8	7	7	0	1	1.00	0.88	0.93
23	10	10	8	0	2	1.00	0.80	0.89
24	11	10	10	0	1	1.00	0.91	0.95
25	2	2	2	0	0	1.00	1.00	1.00
26	8	8	7	0	0	1.00	1.00	1.00
27	9	8	7	0	1	1.00	0.88	0.93
28	12	10	10	0	2	1.00	0.83	0.91
29	7	7	7	0	0	1.00	1.00	1.00
30	8	8	7	1	0	0.88	0.88	0.93
<b>Average</b>						<b>0.99</b>	<b>0.94</b>	<b>0.96</b>

The proposed encoder-decoder based model (LGspikeNet) achieved 99.96% accuracy in detecting/identifying spikes, but counting accuracy is about 94% (Table 4.8). This may be due to undercounting of spikes that overlap each other and object linking (or connecting) problem (Fig 4.31) and invisibility of some spikes which are hidden behind other plant structures. As flood-fill technique employs object count by

growing through similar pixel regions from the starting pixel therefore; if multiple objects are linked together, it will be counted as one object. In this study, an attempt has been made to resolve this problem by taking the images from three different angles i.e.,  $0^0$ ,  $120^0$  and  $240^0$ .



**Fig 4.31** (a) Overlapping problem, (b) Object connecting problem

#### 4.3.3 Online software for spike identification and counting:

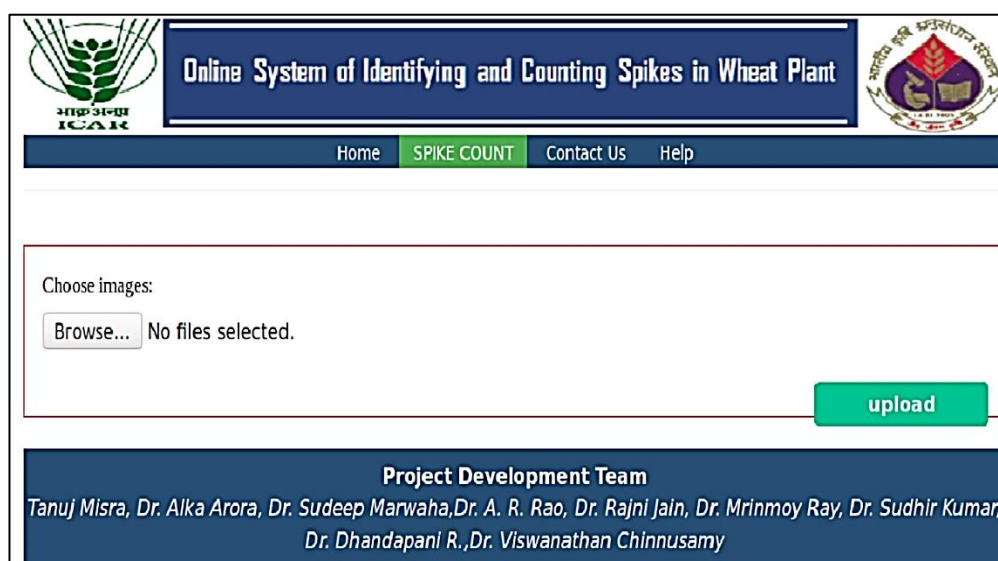
In this study, online software for spike identification and counting has been developed. Details architecture of the software and technologies used have been discussed in the previous chapter. Client Side Interface Layer (CSIL) of the software is responsible for taking input from the user. It has been implemented using Hyper Text Markup language (HTML), Cascading Style Sheet (CSS) and JavaScript. Server Side Application Layer (SSAL) consists of FLASK web development module, deep learning module for spike identification and counting module for counting number of spikes. FLASK web development module is responsible for taking request from CSIL and gives appropriate response to the layer. Deep learning module is built with Tensorflow, keras framework and several python libraries Numpy, Scipy, Matplotlib *etc.* whereas counting module built with “*analyse particles*” function of imageJ.

For identifying and counting wheat spikes in the online software, following steps need to be taken:

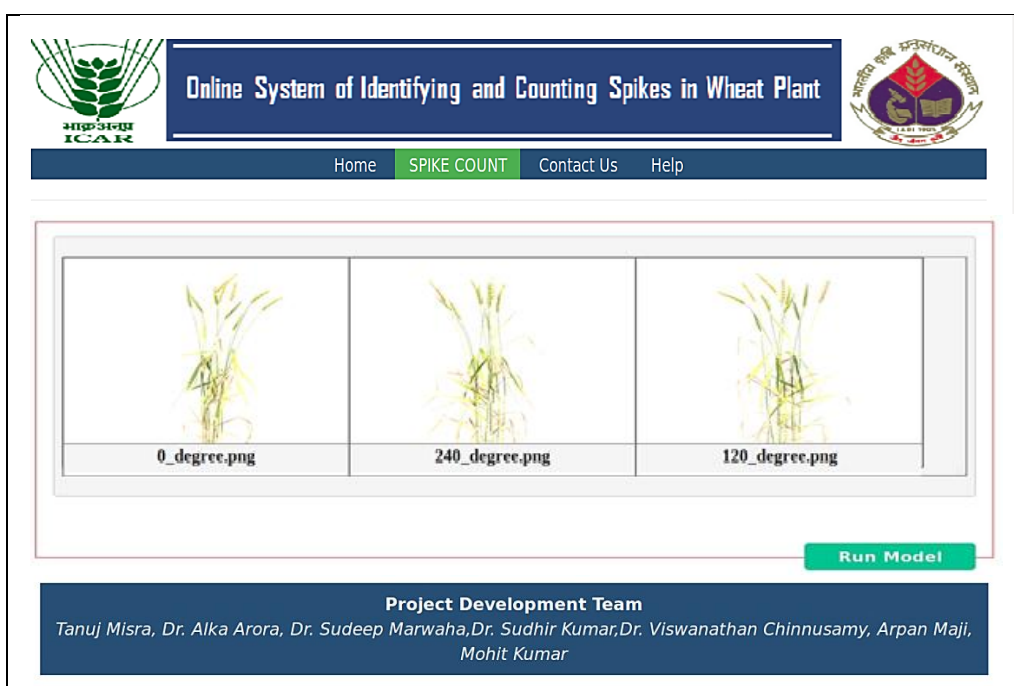
*Step 1:* Browse and select three input images of wheat plant of three directions ( $0^0$ ,  $120^0$ ,  $240^0$ ) and upload them by clicking on “**Upload**” button. Snapshot of the software is given in Fig 4.27.

**Step 2: (Spike identification)** click on “**Run Model**” to apply the developed deep-learning model on the input image for identifying spikes (Fig 4.24). The output image is mask/binary image consists of spikes only (Fig 4.28).

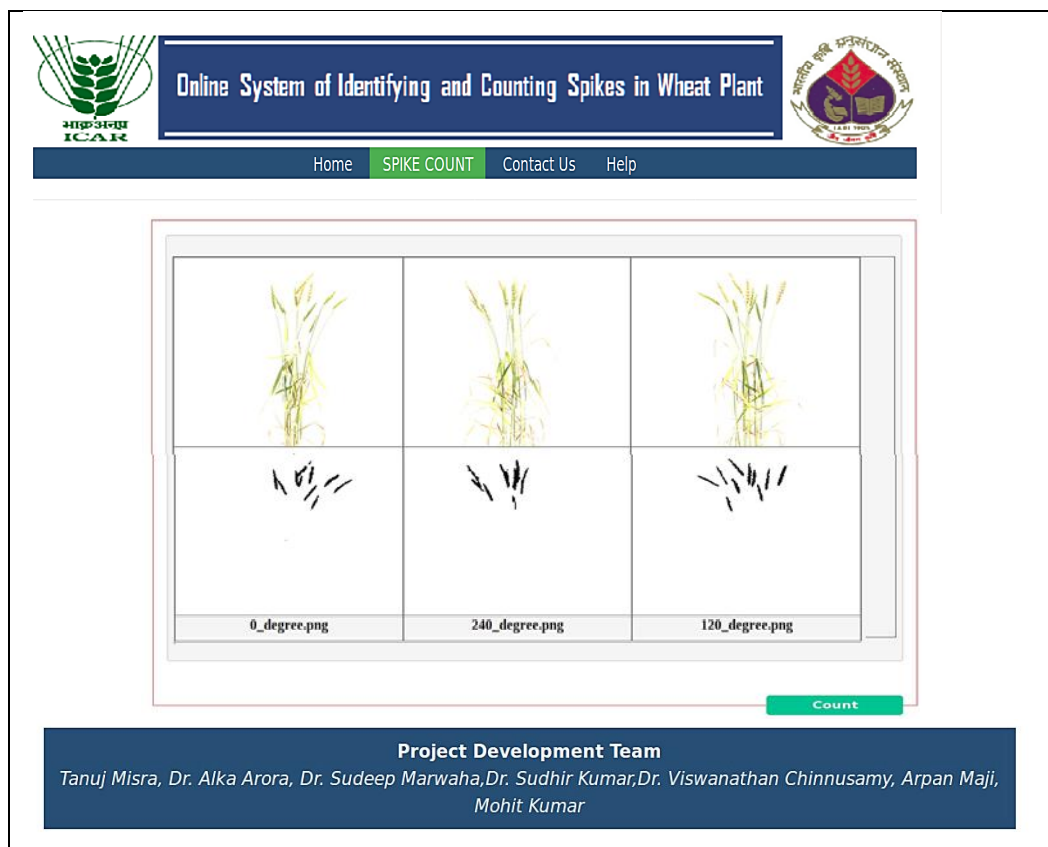
**Step 3: (Spike counting)** click on “Count” button to apply “*analyse particles*” function of imageJ to count number of spikes for each image of three directions. (Fig 4.29).



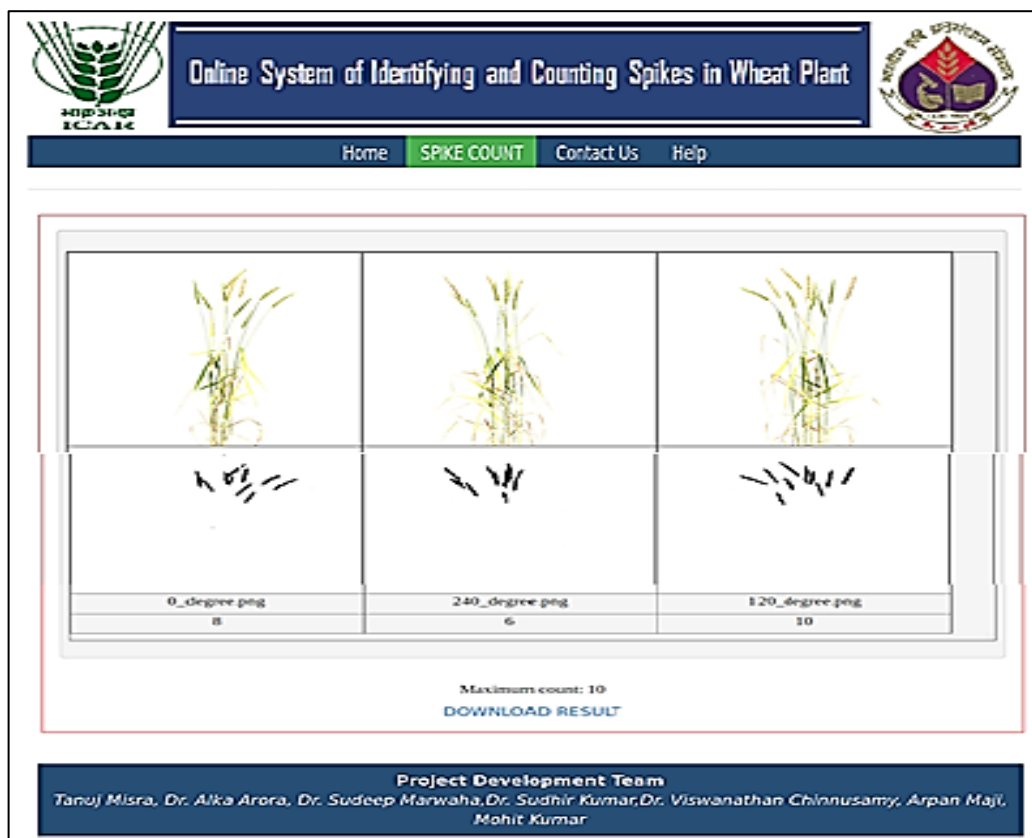
**Fig 4.27** Browse and upload the input images



**Fig 4.28** Click on Run Model for spike identification



**Fig 4.29** Output of spike identification



**Fig 4.30** Output of spike counting

In this study, a new approach has been proposed to estimate LFW by using ANN model with combination of VIS-NIR images. The main feature of the proposed approach is that the concept of moisture content has been included which is the most important basis of LFW. This study uncovered that GLP from VIS image and Mean Gray Value from NIR image can be effectively used for estimating LFW. Another approach has been presented in this study to detect and count wheat spikes in non-destructive manner based on combine effort of digital image analysis and deep learning technique. In this study, two deep learning models have been developed for spike identification namely **SpikeSegNet** (Spike Segmentation Network) and **LGspikeNet** (Local patch extraction and Global mask refinement spike detection Network) based on convolutional encoder-decoder deep learning technique. It was observed that **SpikeSegNet** resulted in some blurriness and irregular shape of the object. To improve upon the same **LGspikeNet** has been proposed for spike identification at local patch level. Online software has also been developed to automate the pipeline of spike detection and counting. The next chapter (Chapter V) contains the summary and conclusion of the whole study.

## CHAPTER V

### SUMMARY AND CONCLUSION

---

Significantly improved crop varieties are much needed to deal with rapidly growing human population scenarios. Besides, systematic quantification of plant phenotypic traits or components in a high-throughput and non-destructive manner is a challenging task. In this context, high-throughput image analysis for plant phenotyping can be used to extract several phenotypic parameters related to plant growth, physiology, yield *etc.* avoiding time consuming manual efforts. In this study, rice and wheat crop have been selected because of their importance as major food crop and vast available background knowledge of their physiology, genetics and genomics. New approaches have been proposed based on image analysis and machine learning technique to derive phenotypic traits like biomass in rice plant and spike identification and counting in wheat plant.

Plant biomass plays a vital role in studying functional plant biology and growth analysis. It is an important factor to determine plant growth rate and net primary production. Leaf Fresh Weight (LFW) is one of the parameters used to estimate the plant biomass. Conventional method of estimating LFW is not only time consuming and laborious but also destructive in the nature, Some studies are available for estimating plant biomass through non-destructive image analysis technique (Paruelo *et al.*, 2000; Mizoue and Masutani 2003; Golzarian *et al.*, 2011; Schirrmann *et al.*, 2016), but most of them considered projected shoot area as a linear function of plant biomass. None of them, considered water content which is the key component of biomass. In this study, it is hypothesized that combined use of visual (VIS) and near infra-red (NIR) image can compute plant biomass more precisely than VIS image only as NIR reflectance image is used to measure water content of the plant. For developing the model of LFW estimation in rice plant through image analysis, VIS and NIR images are taken from high-throughput plant phenotyping facility established at Nanaji Deshmukh Plant Phenomics Centre, ICAR-IARI, New Delhi. Two image derived parameter *i.e.*, Green Leaf Proportion (GLP) from VIS image and Mean Gray Intensity (NIR\_MGI) from NIR images have been used for building the machine learning model to estimate LFW. The proposed

approach is named as **VN\_LFW**. Artificial Neural Network (ANN) technique has been used in model development and its performance has been compared with linear regression technique. The image dataset (104 samples) has been divided into training and testing at 85:15 ratios to develop ANN model to estimate LFW. Distinctive combinations of hidden layer and hidden nodes in each layer has been attempted and out of which one hidden layers with 5 hidden nodes performed superior than other combinations. Thus, the ANN architecture for estimating LFW consists of one input layer with two input nodes (GLP and NIR\_MGI), one hidden layers with 5 hidden nodes and one output layer with one output node (LFW). The proposed approach has been compared with the conventional image processing based approach (linear function of projected shoot area from VIS image only) as well as linear regression approach based on GLP from VIS image and NIR\_MGI from NIR image. Proposed approach VN\_LFW has outperformed than the other comparative approaches in both train and test dataset with Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) as 0.15 and 9.55 in training dataset and 0.13 and 9.65 in testing dataset respectively. The algorithm of measuring GLP and NIR\_MGI has been implemented in Matlab software.

Another image processing based approach has been given for spike identification and counting in wheat plant Spike counting per plant or per unit area through naked-eye is a laborious, destructive and time consuming process for large amount of genotypes. Yield estimation of wheat in high-throughput and non-destructive manner has received a significant research attention. In this context, few literatures (Bi *et al.*, 2010; Duan *et al.*, 2015; Zhao *et al.*, 2015; Li *et al.*, 2017; Sadeghi-Tehran *et al.*, 2017; Pound *et al.*, 2017; Hasan *et al.*, 2018) are available in the area of spike/panicle detection and characterization through image analysis. In some cases, images were captured after cutting spikes from the plant which is destructive in nature. Some researchers applied machine learning techniques by manually defining colour intensity and texture component for segmenting the spike regions. In the recent trend, it has been seen that computer visions particularly object detection plays an important role in non-destructive plant phenotyping through digital image analysis which is being helpful for automatic detection and counting of spikes in wheat plant (Pound *et al.*, 2017; Hasan *et al.*, 2018). In this context, an approach has been presented based on combined effort of digital image analysis and

deep learning techniques which involve identification and counting of spikes from the digital images of whole wheat plant.

For spike identification, deep learning models have been developed namely **SpikeSegNet** (Spike Segmentation Network) and **LGspikeNet** (Local patch extraction and Global mask refinement Spike detection Network) based on convolutional encoder-decoder deep learning technique, whereas for counting number of spikes per plant, “*analyse particles*” function of imageJ which implements flood-fill image analysis technique has been applied on the output image (binary/mask image containing spike regions only) of the developed model. For developing the deep learning model and validating the proposed approach, VIS images of wheat plants are taken from three different angles ( $0^\circ$ ,  $120^\circ$ ,  $240^\circ$ ) of the plant. To develop the model, image dataset of 200 plants with 3 images each are split randomly into training and validation at 85:15 ratios. The developed model was validated on the validation dataset (images of randomly selected 15% of plants). The proposed **SpikeSegNet** network model architecture consists of 3 encoder blocks and corresponding hierarchy of 3 decoder blocks and the network has been trained end-to-end by resizing the original image (of size 1656\*1356) into 256\*256. Encoder blocks take input image to give feature map representation that holds the contextual and spatial information and decoder blocks take the information as input and produce corresponding segmentation masks as output. For spike identification, precision, accuracy and robustness ( $F_1$  score) of the proposed **SpikeSegNet** network model has been found as 94.56, 94.66 and 94.88% respectively. It was observed that, **SpikeSegNet** network results some sort of irregular and inaccurate segmentation of the objects (or, spike). As the network has been trained end-to-end by resizing the original image into 256\*256, it loses the contextual as well as spatial information which is very essential to learn features for the segmentation network. To improve upon the same, another deep network **LGspikeNet** has been proposed. **LGspikeNet** network model architecture is a combination of two especial feature networks namely **Local Patch Extraction (LPspikeNet)** and **Global Mask Refinement (GMspikeNet)**. The network has been trained at patch level. Patches are nothing but the small size parts of an image. For spike identification, precision, accuracy and robustness ( $F_1$  score) of the proposed **LGspikeNet** network model has been found as 99.95, 99.96 and 99.96% respectively, while for spike counting the precision,

accuracy and robustness were 99, 94 and 92% respectively. Although the proposed encoder-decoder model for detecting/identifying spikes achieved 99.96% accuracy but counting accuracy is comparatively less (94%). This is because of overlapping of spikes and invisibility of some spikes which are hidden behind other plant structures. In this thesis, online software for identification and counting of wheat spikes has also been developed by using the proposed **LGspikeNet** network model. Client Side Interface Layer (CSIL) of the developed software has been implemented using Hyper Text Markup language (HTML), Cascading Style Sheet (CSS) and JavaScript technology. Server Side Application Layer (SSAL) has been built using Flask web-development tool for taking the input from users and Deep learning module built with Tensorflow, Keras framework and Numpy, Scipy, Matplotlib, OpenCV *etc.* libraries of python for detection of spikes and ImageJ module has been integrated to count the objects (or, spikes) on the output image of the deep learning module.

Thus, in this study a new approach, VN\_LFW, has been proposed to estimate LFW by using ANN model with combination of VIS-NIR images. The main feature of the proposed approach is that the concept of moisture content has been included which is the most important basis of measuring LFW. This study also uncovers that GLP from visual image and NIR\_MGI from NIR image can be effectively used for estimating LFW and the developed model out performs over projected shoot area based conventional approach as well as the linear regression approach based on GLP and NIR\_MGI. Another approach has been presented in this study to detect and count wheat spikes in non-destructive manner based on combine effort of digital image analysis and deep learning technique. Online software has also been developed to automate the pipeline of spike detection and counting. The approach can be applicable to predict yield of the particular plant by using the spike count and pixel area of the spikes as well as the presence of chaffy grain in the spike can be detected by differences in moisture content which can be obtained by superimposing the output of LGspikeNet network (*i.e.*, binary image consists of spike regions only) on the NIR image. This approach may be useful to other cereal crops like rice, maize in future.

## ABSTRACT

---

Quantification of phenotypic parameter is necessary to meet the future demand of agricultural production. Conventional measurements of these traits/parameters are time-consuming, destructive and labour-intensive. In this study, new approaches have been proposed based on image analysis and machine learning technique to derive phenotypic traits like Leaf Fresh Weight (LFW) in rice plant and spike identification and counting in wheat plant. For this purpose, images have been taken from high-throughput plant phenotyping facility established at Nanaji Deshmukh Plant Phenomics Centre, ICAR-IARI, New Delhi. In this study, it is hypothesized that combined use of visual (VIS) and near infra-red (NIR) image can compute LFW more precisely than VIS image only as NIR reflectance image is used to measure water content of the plant. Two image derived parameters *i.e.*, Green Leaf Proportion (GLP) from VIS image and Mean Gray Intensity (NIR\_MGI) from NIR images have been used for building Artificial Neural Network (ANN) model to estimate LFW. The proposed approach is named as VN\_LFW. The proposed approach significantly enhanced the fresh biomass prediction as compared with the conventional regression technique in both train and test dataset with Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) as 0.15 and 9.55 in training dataset and 0.13 and 9.65 in testing dataset respectively. The algorithm of measuring GLP and NIR\_MGI has been proposed and the macro has been developed using Matlab software. Another significant area for spike identification has been attempted with deep learning models of Artificial Intelligence. Two models have been developed for spike identification namely **SpikeSegNet** (Spike Segmentation Network) and **LGspikeNet** (Local patch extraction and Global mask refinement Spike detection Network) based on convolutional encoder-decoder deep learning technique. For counting number of spikes per plant, “*analyse particles*” function of imageJ which implements flood-fill image analysis technique has been applied on the output image (binary/mask image containing spike regions only) of the developed model. For spike identification, precision, accuracy and robustness ( $F_1$  score) of the proposed **SpikeSegNet** model has been found as 94.56, 94.66 and 94.88% respectively whereas for **LGspikeNet** it has been as 99.95, 99.96 and 99.96% respectively. In spike counting using **LGspikeNet**, the metric values are 99, 94 and 92% respectively. Online software for identification and counting of wheat spikes has also been developed by using the proposed **LGspikeNet** network model.

**Keywords:** deep learning, encoder-decoder deep network, GLP, image analysis, LFW, moisture content, NIR, NIR\_MGI, non-destructive plant phenotyping, rice, VIS, wheat spikes identification and count.

## सार

कृषि उत्पादन की भविष्य की मांग को पूरा करने के लिए फैनोटिपिक प्राचलों की मात्रा का निर्धारण आवश्यक है। इन लक्षणों/प्राचलों के पारम्परिक माप समय लेने वाले, डिस्ट्रक्टिव एवं श्रम-गहन हैं। इस अध्ययन में, चावल के पौधे में लीफ फ्रैष व्हेट (LFW) जैसे फैनोटिपिक लक्षणों को प्राप्त करने तथा गेहूँ के पौधे में स्पाइक की पहचान एवं गणना करने के लिए इमेज एनालिसिस तथा मशीन लर्निंग टेक्नीक के आधार पर नयी अप्रोच प्रस्तावित की गयी हैं। इस उद्देश्य के लिए, इमेजेज नानाजी देशमुख प्लांट फिनोमिक्स सेंटर, भा. कृ.अ.प.-भा.कृ.अ.सं., नई दिल्ली में स्थापित हाई-थ्रूपुट प्लांट फिनोटिपिंग फेसिलिटी से प्राप्त की गई। इस अध्ययन में, यह परिकल्पना की गई है कि विजुअल (VIS) एवं नियर इन्फ्रा-रेड (NIR) के संयुक्त उपयोग से केवल VIS इमेज की अपेक्षा LFW अधिक सटीकता से गणना कर सकता है क्योंकि एनआईआर रिफ्लेक्टेंस इमेज का उपयोग पौधे की जल की मात्रा को मापने के लिये किया जाता है। LFW के आकलन के लिये आर्टिफिशियल न्यूरल नेटवर्क (ANN) मॉडल के निर्माण के लिये दो इमेज व्युत्पन्न प्राचलों अर्थात् वीआईएस इमेज से ग्रीन लीफ प्रपोरषन (GLP) तथा एनआईआर इमेजेज से मीन ग्रे इन्टेनसिटी (NIR\_MGI) का प्रयोग किया गया है। प्रस्तावित अप्रोच को VN\_LFW का नाम दिया गया है। प्रस्तावित अप्रोच ने रूट मीन स्क्वेअर एरर (RMSE) एवं मीन एब्सोल्यूट परसेन्टेज एरर (MAPE) के साथ टेस्टिंग डाटा सैट में क्रमशः 0.15 एवं 9.55 तथा 0.13 एवं 9.65 के रूप में ट्रेन एवं टेस्ट डाटा सैट दोनों में पारम्परिक समाश्रयण तकनीक की तुलना में फ्रैष बायोमास प्रिडिक्शन को काफी बढ़ाया। GLP एवं NIR\_MGI को मापने के लिये एल्गोरिथ्म का प्रस्ताव दिया गया है तथा मैटलैब (Matlab) सॉफ्टवेयर का उपयोग करते हुये मैक्रो विकसित किया गया। स्पाइक पहचान के लिये एक और महत्वपूर्ण क्षेत्र आर्टिफिशियल इन्टेलिजेन्स के गहन लर्निंग मॉडलों के साथ प्रयास किया गया। स्पाइक पहचान के लिये दो मॉडल, नामतः SpikeSegNet (Spike Segmentation Network) एवं LGspikeNet (Local patch extraction and Global mask refinement Spike detection Network) विकसित किये गये हैं जो कनवोल्यूषनल एनकोडल-डिकोडर डीप लर्निंग टेक्नीक पर आधारित हैं। प्रति पौधा स्पाइक्स की गणना करने के लिये इमेजJ के “एनालाइज पार्टिकल्स” फलन, जो फल्ट-फिल इमेज एनालिसिस तकनीक को क्रियान्वित करता है, विकसित मॉडल की आउटपुट इमेज (बाइनरी-मासक इमेज युक्त स्पाइक क्षेत्र केवल) पर एप्लाइ किया गया है। प्रस्तावित SpikeSegNet मॉडल स्पाइक पहचान के लिये यथार्थता, सटीकता एवं रॉबस्टनेस ( $F_1$  score) क्रमशः 94.56, 94.66 एवं 94.88 प्रतिशत पाई गई जबकि LGspikeNet के लिये यह क्रमशः 99.95, 99.96 एवं 99.96 थी। LGspikeNet का प्रयोग करते हुये मैट्रिक मान क्रमशः 99, 94 एवं 92 प्रतिशत थे। गेहूँ के स्पाइक्स की पहचान एवं पहचान एवं गणना के लिये प्रस्तावित LGspikeNet नेटवर्क मॉडल का उपयोग करते हुये ऑनलाइन सॉफ्टवेयर भी विकसित किया गया।

## REFERENCES

---

- Adamsen, F. G., Pinter, P. J., Barnes, E. M., LaMorte, R. L., Wall, G. W., Leavitt, S. W., Kimball, B. A. (1999).** Measuring wheat senescence with a digital camera. *Crop Science*, 39(3), 719-724.
- Agostinelli, F., Hoffman, M., Sadowski, P., Baldi, P. (2014).** Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.
- Asundi, A., & Wensen, Z. (1998).** Fast phase-unwrapping algorithm based on a gray-scale mask and flood fill. *Applied optics*, 37(23), 5416-5420.
- Berners-Lee, T. (1990).** Information Management: A Proposal. CERN. <http://www.w3.org>.
- Bi, K., Jiang, P., Li, L., Shi, B., Wang, C. (2010).** Non-destructive measurement of wheat spike characteristics based on morphological image processing. *Transactions of the Chinese Society of Agricultural Engineering*, 26(12), 212-216.
- Bognár, P., Kern, A., Pásztor, S., Lichtenberger, J., Koronczay, D., Ferencz, C. (2017).** Yield estimation and forecasting for winter wheat in Hungary using time series of MODIS data. *International journal of remote sensing*, 38(11), 3394-3414.
- Bognár, P., Kern, A., Pásztor, S., Lichtenberger, J., Koronczay, D., Ferencz, C. (2017).** Yield estimation and forecasting for winter wheat in Hungary using time series of MODIS data. *International journal of remote sensing*, 38(11), 3394-3414.
- Bruinsma, J. (2003).** World Agriculture Towards 2015/2030” an FAO perspective. Earthscan, London.
- Burns, J. and Growney, A. S. (2001).** JavaScript Goodies. Pearson Education. New Jersey.
- Chaohul, L & Hui, R. (2010).** Leaf area measurement based on image processing. *IEEE*, 580-582.

- Duan, L., Huang, C., Chen, G., Xiong, L., Liu, Q., Yang, W. (2015).** Determination of rice panicle numbers during heading by multi-angle imaging. *The Crop Journal*, 3(3), 211-219.
- Dunne, R. A., & Campbell, N. A. (1997).** On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, 181, 185.
- Fernández, R., Montes, H., Salinas, C. (2015).** VIS-NIR, SWIR and LWIR imagery for estimation of ground bearing capacity. *Sensors*, 15(6), 13994-14015.
- Finkel, E. (2009).** With ‘phenomics,’ plant scientists hope to shift breeding into overdrive. *Science*, 325: 380-381.
- Furbank, R. T., von Caemmerer, S., Sheehy, J., Edwards, G. (2009).** C4 rice: a challenge for plant phenomics. *Functional Plant Biology*, 36(11), 845-856.
- Gehan, M. A., Fahlgren, N., Abbasi, A., Berry, J. C., Callen, S. T., Chavez, L. & Hoyer, J. S. (2017).** PlantCV v2: Image analysis software for high-throughput plant phenotyping. *PeerJ*, 5, e4088.
- Girshick, R. (2015).** “Fast r-cnn,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV ’15 (Washington, DC: IEEE Computer Society), 1440–1448. doi: 10.1109/ICCV. 2015.169
- Golzarian, M. R., Frick, R. A., Rajendran, K., Berger, B., Roy, S., Tester, M., Lun, D. S. (2011).** Accurate inference of shoot biomass from high-throughput images of cereal plants. *Plant methods*, 7(1), 2.
- Hagenauer, J., Offer, E., Papke, L. (1996).** Iterative decoding of binary block and convolutional codes. *IEEE Transactions on information theory*, 42(2), 429-445.
- Haindl, M. & Krupička, M. (2015).** Unsupervised detection of non-iris occlusions. *Pattern Recognition Letters*, 57, 60-65.
- Hasan, M. M., Chopin, J. P., Laga, H., Miklavcic, S. J. (2018).** Detection and analysis of wheat spikes using Convolutional Neural Networks. *Plant methods*, 14(1), 100.

**He, K., Zhang, X., Ren, S., Sun, J. (2016).** “Deep residual learning for image recognition,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV).

<https://www.plantphenomics.org.au/>

<https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55>

[https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)

<https://medium.com/@aggirma/part-1-convolutional-neural-network-in-a-nutshell-89f81a329ec3>

<https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694>

[https://github.com/aleju/papers/blob/master/neural-nets/Batch\\_Normalization.md](https://github.com/aleju/papers/blob/master/neural-nets/Batch_Normalization.md)

<https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23>

<https://www.tutorialspoint.com/tensorflow>

[https://www.tutorialspoint.com/tensorflow/tensorflow\\_keras](https://www.tutorialspoint.com/tensorflow/tensorflow_keras)

<http://rsbweb.nih.gov/ij/docs/>

<http://rsbweb.nih.gov/ij/developer/api/>

<https://en.wikipedia.org/wiki/Flask>

<https://en.wikipedia.org/wiki/PyCharm>

<https://www.jetbrains.com/pycharm/>

<https://www.eweek.com/development/jetbrains-strikes-python-developers-with-pycharm-1.0-ide>

**Huang, T., Yang, G. J. T. G. Y., Tang, G. (1979).** A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1), 13-18.

- Huertas, A. & Medioni, G. (1986).** Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5), 651-664.
- Hunt Jr, E. R. & Rock, B. N. (1989).** Detection of changes in leaf water content using near-and middle-infrared reflectances. *Remote sensing of environment*, 30(1), 43-54.
- Ioffe, S. & Szegedy, C. (2015).** Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- Jang, E., Gu, S., & Poole, B. (2016).** Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.
- Jansen, M., Gilmer, F., Biskup, B., Nagel, K. A., Rascher, U., Fischbach, A., Briem, S., Dreissen, G., Tittmann, S., Braun, S., De Jaeger, I. (2009).** Simultaneous phenotyping of leaf growth and chlorophyll fluorescence via GROWSCREEN FLUORO allows detection of stress tolerance in *Arabidopsis thaliana* and other rosette plants. *Functional Plant Biology*, 36(11), 902-914.
- Jaswal, G., Jha, R. R., Gupta, D., Saini, S. (2019).** PixISegNet: Pixel Level Iris Segmentation Network using Convolutional Encoder-Decoder with Stacked Hourglass Bottleneck. *IET Biometrics*.
- Jin, X., Liu, S., Baret, F., Hemerlé, M. and Comar, A. (2017).** Estimates of plant density of wheat crops at emergence from very low altitude uav imagery. *Remote Sens. Environ.* 198, 105–114. doi: 10.1016/j.rse.2017.06.007
- Kanopoulos, N., Vasanthavada, N., Baker, R. L. (1988).** Design of an image edge detection filter using the Sobel operator. *IEEE Journal of solid-state circuits*, 23(2), 358-367.
- Kawashima, S. and Nakatani M. (1998).** An algorithm for estimating chlorophyll content in leaves using a video camera. *Annals of Botany*, 81:49-54.
- Kingma, D. P., & Ba, J. (2014).** Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

- Klukas, C., Chen, D., Pape, J. M. (2014).** Integrated analysis platform: an open-source information system for high-throughput plant phenotyping. *Plant physiology*, 165(2), 506-518.
- Knecht, A. C., Campbell, M. T., Caprez, A., Swanson, D. R., Walia, H. (2016).** Image Harvest: an open-source platform for high-throughput plant image processing and analysis. *Journal of experimental botany*, 67(11), 3587-3599.
- Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012).** Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems*. 1097-1105.
- Lim, K. S. & Treitz, P. M. (2004).** Estimation of above ground forest biomass from airborne discrete return laser scanner data using canopy-based quantile estimators. *Scandinavian Journal of Forest Research*, 19(6), 558-570.
- Li, Q., Cai, J., Berger, B., Okamoto, M., Miklavcic, S.J. (2017).** Detecting spikes of wheat plants using neural networks with laws texture energy. *Plant Methods*. 13:1–13.
- Li, Z., Ji, C. and Liu, J. (2008).** Leaf Area Calculating Based on Digital Image. *Computer and Computing Technologies in Agriculture*, 259: 1427-33.
- Lukina, E. V., Stone, M. L., Raun, W. R. (1999).** Estimating vegetation coverage in wheat using digital images. *Journal of Plant Nutrition*, 22(2), 341-350.
- Mahdi, M. A., Ahmed, Al-Ani, Derek, E. and Daniel, K.Y.T. (2012).** A New Image Processing Based Technique to Determine Chlorophyll in Plants. *American-Eurasian J. Agric. and Environ. Sci.*, 12(10): 1323-1328, 2012.
- Mao, X., Shen, C. & Yang, Y. B. (2016).** Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *In Advances in neural information processing systems*. 2802-2810.
- Marcon, M. and Mariano, K. (2011).** Estimation of total leaf area in perennial plants using image analysis. *R. Bras. Eng. Ambiental*, 15: 96-101.

- Misra, T., Priyadarshini, S., Arora, A., Marwaha, S., Roy, H. S., & Ray, M. (2018).** A comparative study of chlorophyll content estimation techniques through image analysis. *Journal of Crop and Weed*, 14(3), 165-168.
- Mizoue, N. & Masutani, T. (2003).** Image analysis measure of crown condition, foliage biomass and stem growth relationships of *Chamaecyparis obtusa*. *Forest Ecology and Management*, 172(1), 79-88.
- Moghaddam, P. A., Derafshi, M.H. and Shirzad, V. (2011).** Estimation of single leaf chlorophyll content in sugar beet using machine vision. *Turk J Agric For*.35: 563-568.
- Mohanty, S. P., Hughes, D. P. & Salathé, M. (2016).** Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7, 1419.
- Montes, N., Gauquelin, T., Badri, W., Bertaudiere, V., & Zaoui, E. H. (2000).** A non-destructive method for estimating above-ground forest biomass in threatened woodlands. *Forest Ecology and Management*, 130(1-3), 37-46.
- Mora, C., Tittensor, D. P., Adl, S., Simpson, A. G., & Worm, B. (2011).** How many species are there on Earth and in the ocean?. *PLoS biology*, 9(8).
- Niklas, K. J. & Enquist, B. J. (2002).** On the vegetative biomass partitioning of seed plant leaves, stems, and roots. *The American Naturalist*, 159(5), 482-497.
- Olabode, O., & Olabode, B. T. (2012).** Cerebrovascular accident attack classification using multilayer feed forward artificial neural network with back propagation error.
- Otsu, N. (1979).** A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1), 62-66.
- Paruelo, J. M., Lauenroth, W. K. & Roset, P. A. (2000).** Estimating aboveground plant biomass using a photographic technique. *Journal of Range Management*, 190-193.
- Patil, S. B. and Bodhe, S.K. (2011).** Betel Leaf Area Measurement Using Image Processing. *International Journal on Computer Science and Engineering (IJCSE)*. 3, 7.

- Porra, R. J., Thompson, W. A. and Kreidemann, P. E. (1989).** Determination of accurate extinctions coefficients and simultaneous equations for assaying chlorophylls a and b extracted with four different solvents: verification of concentration of chlorophyll standards by atomic absorption spectroscopy. *Biochim Biophys Acta*, 975: 384-94.
- Poorter, H. & Nagel, O. (2000).** The role of biomass allocation in the growth response of plants to different levels of light, CO<sub>2</sub>, nutrients and water: a quantitative review. *Functional Plant Biology*, 27(12), 1191-1191.
- Pound, M. P., Atkinson, J. A., Wells, D. M., Pridmore, T. P. & French, A. P. (2017).** Deep learning for multi-task plant phenotyping. In *Proceedings of the IEEE International Conference on Computer Vision*, 2055-2063.
- Proenca, H., Filipe, S., Santos, R., Oliveira, J. & Alexandre, L. A. (2010).** The ubiris. v2: A database of visible wavelength iris images captured on-the-move and at-a-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8), 1529-1535.
- Qiongyan, L., Cai, J., Berger, B., Okamoto, M., Miklavcic, S. J. (2017).** Detecting spikes of wheat plants using neural networks with Laws texture energy. *Plant methods*, 13(1), 83.
- Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., Hughes, D. P. (2017).** Deep learning for image-based cassava disease detection. *Frontiers in plant science*, 8, 1852.
- Ronneberger, O., Fischer, P. & Brox, T. (2015).** U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. 234-241.
- Sadeghi-Tehran, P., Sabermanesh, K., Virlet, N., Hawkesford, M. J. (2017).** Automated method to determine two critical growth stages of wheat: heading and flowering. *Frontiers in plant science*, 8, 252.
- Schirrmann, M., Hamdorf, A., Garz, A., Ustyuzhanin, A., Dammer, K. H. (2016).** Estimating wheat biomass by combining image clustering with crop height. *Computers and Electronics in Agriculture*, 121, 374-384.

- Schneider, C. A., Rasband, W. S., & Eliceiri, K. W. (2012).** NIH Image to ImageJ: 25 years of image analysis. *Nature methods*, 9(7), 671-675.
- Seber, G. A., & Lee, A. J. (2012).** Linear regression analysis (Vol. 329). John Wiley & Sons.
- Seelig, H. D., Hoehn, A., Stodieck, L. S., Klaus, D. M., Adams Iii, W. W., Emery, W. J. (2008).** The assessment of leaf water content using leaf reflectance ratios in the visible, near, and shortwave infrared. *International Journal of Remote Sensing*, 29(13), 3701-3713.
- Serrano, L., Ustin, S. L., Roberts, D. A., Gamon, J. A., Penuelas, J. (2000).** Deriving water content of chaparral vegetation from AVIRIS data. *Remote sensing of Environment*, 74(3), 570-581.
- Sestak, Z. and Catsky, J. (1971).** *Plant Photosynthesis Production. Manual of Methods*, Junk, Netherlands.
- Simonyan, K. & Zisserman, A. (2014).** Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., Stefanovic, D. (2016).** Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, 2016.
- Smith, S. M., Garrett, P. B., Leeds, J. A., McCormick, P. V. (2000).** Evaluation of digital photography for estimating live and dead aboveground biomass in monospecific macrophyte stands. *Aquatic Botany*, 67(1), 69-77.
- Soule, M. (1967).** Phenetics of natural populations. II. Asymmetry and evolution in a lizard. *The American Naturalist*, 101(918), 141-160.
- Stenberg, B., Rossel, R. A. V., Mouazen, A. M., Wetterlind, J. (2010).** Visible and near infrared spectroscopy in soil science. *In Advances in agronomy*, 107, 163-215.
- Sticklen, M. B. (2007).** Feedstock crop genetic engineering for alcohol fuels. *Crop science*, 47(6), 2238-2248.

- Subramaniam<sup>1</sup>, A., Carpenter, E. J., & Falkowski, P. G. (1999).** Bio-optical properties of the marine diazotrophic cyanobacteria *Trichodesmium* spp. II. A reflectance model for remote sensing. *Limnology and Oceanography*, 44(3), 618-627.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015).** Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 1-9.
- Teodoro, A. C. (2015).** Applicability of data mining algorithms in the identification of beach features/patterns on high-resolution satellite data. *Journal of Applied Remote Sensing*, 9(1), 095095.
- Tsaftaris, S. A., Minervini, M., Scharr, H. (2016).** Machine learning for plant phenotyping needs image processing. *Trends in plant science*, 21(12), 989-991.
- Xiong, X., Duan, L., Liu, L., Tu, H., Yang, P., Wu, D., Chen, G., Xiong, L., Yang, W., Liu, Q. (2017).** Panicle-SEG: a robust image segmentation method for rice panicles in the field based on deep learning and superpixel optimization. *Plant methods*, 13(1), 104.
- Yuzhu, H., Xiaomei, W. and Shuyao, S. (2011).** Nitrogen determination in pepper (*Capsicum frutescens* L) by colour image analysis (RGB). *African Journal of Biotechnology*, 77: 17737-41.
- Zhao, Z. & Ajay, K. (2015).** An accurate iris segmentation framework under relaxed imaging constraints using total variation model. *In Proceedings of the IEEE International Conference on Computer Vision*, 3828-3836.

## APPENDIX I

---

### ❖ MATLAB code of GLP computation from VIS image:

```
clc;
clear all;
close all;
imtool close all;
tic;
I=imread('D:\Raju Sir\RWC\NIR_R2_06112016_NIR_RWC_1_2016-11-06_06-51-14_4346100\VIS_SV\0_0_0.png');
I2=imcrop(I,[1358.5 1830.5 396 816]);%cropped image
r=double(I2(:,:,1));
g=double(I2(:,:,2));
b=I2(:,:,3);
% [a1,b1,c1]=size(I2);
ndi=((g-r)./(g+r));%actual NDI image
% ndigray=mat2gray(ndi);%normalize of NDI image
imtool(ndi);
imtool(mat2gray(ndi))
[a1,b1,c1]=size(I2);
thresh1=multithresh(ndi);
%thresh=multithresh(mat2gray(ndi));
ind=1;%index variable
for i=1:a1
    for j=1:b1
        % if ndi(i,j)==0 %for thresh==0
        % I3(i,j)=0;
        if ndi(i,j)>thresh1%otsu
            I3(i,j)=1;
        else
            %I3(i,j)=1;% thresh==0
            I3(i,j)=0;%otsu's threshold
            % ndi1(ind)=ndi(i,j);
            % ind=ind+1;
        end
    end
end
imtool(I3)

r2=I3.*double(I2(:,:,1));
g2=I3.*double(I2(:,:,2));
b2=I3.*double(I2(:,:,3));

I4=cat(3,r2,g2,b2);

for i=1:a1
    for j=1:b1
```

```

if I4(i,j,2)>I4(i,j,1)&& I4(i,j,2)>I4(i,j,3)
    % I5(i,j)=1;
    I5(i,j)=I2(i,j);
    ndi1(ind)=I2(i,j,1);
    ndi2(ind)=I2(i,j,2);
    ndi3(ind)=I2(i,j,3);
    ind=ind+1;
else
    I5(i,j)=0;
end
end
end
imtool(I5);
Area=size(ndi2);
me=mean(ndi2); %mean of projected area
mo=mode(ndi2); %mode of projected area
med=median(ndi2); %median of projected area
% std=std(ndi1);

toc;

```

❖ **MATLAB code of NIR\_MGI computation from NIR image:**

```

clc;
clear All;
imtool close all;
tic;
I=imread('D:\Raju Sir\RWC\NIR_R2_06112016_NIR_RWC_1_2016-11-06_06-51-14_4346100\NIR_SV\0_0_0.png');
NI=imread('D:\Raju Sir\RWC\NIR_R2_06112016_RWC_1_2016-11-15_12-59-19_4470200\NIR_SV\0_0_0.png');%empty image

I2=imcrop(I,[110.967455621302 203.523668639053 55.6094674556213 285.562130177515]);%cropped image
NI2=imcrop(NI,[110.967455621302 203.523668639053 55.6094674556213 285.562130177515]);%cropped image

imtool(I2);
imtool(NI2);

SubtractImage=NI2-I2;
imtool(SubtractImage);
sub2gray=mat2gray(SubtractImage);
imtool(sub2gray);
r=(sub2gray(:,:,1));

[a,b,c]=size(r);
thresh=multithresh(r);
for i=1:a
    for j=1:b

```

```

        if r(i,j)<thresh
            I3(i,j)=0;
        else
            I3(i,j)=1;
        end
    end
end

imtool(I3);

% laplacian filter of the Original Image (I2)
H = fspecial('laplacian');
% apply laplacian filter.
blurred = imfilter(I2,H);
imtool(blurred);
% subtract of original and blurred
SubtractImageLaplac=I2-blurred;
imtool(SubtractImageLaplac);
% apply sobel operator for detecting edges
BW1 = edge(I2(:,:,1),'sobel');
imtool(BW1);
% apply median filter to sobel image and invert
medianblur = medfilt2(BW1);
invertmedBlur = imcomplement(medianblur);
% imtool(invertmedBlur);
IM2=imadd(blurred(:,:,1),uint8(invertmedBlur));
imtool(IM2);

threshIM2=multithresh(IM2);

for i=1:a
    for j=1:b
        if IM2(i,j)<12
            I4(i,j)=0;
        else
            I4(i,j)=1;
            % SubtractImage_1(ind)=r(i,j);
            % ind=ind+1;
        end
    end
end
imtool(I4);

se= strel('square',2)
erodedI = imerode(I4,se);
imtool(erodedI);

%IM5=imadd(double(BW1),erodedI);
IM5=imadd(double(BW1),I3);%

```

```

imtool(IM5);

ind=1;
for i=1:a
for j=1:b
if IM5(i,j)==2 || IM5(i,j)==1
I44(i,j)=1;
else
I44(i,j)=0;
% OutputImage_1(ind)=r(i,j);
% ind=ind+1;
end
end
end

FinalImage=double(I44).*double(I2(:, :, 1));
imtool(FinalImage);

for i=1:a
for j=1:b
if FinalImage(i,j)==0
OutputImage(i,j)=0;
else
OutputImage(i,j)=FinalImage(i,j);
OutputImage_1(ind)=FinalImage(i,j);
ind=ind+1;
end
end
end

z=mat2gray(FinalImage);
imtool(z);

Area=size(OutputImage_1);
me=mean(OutputImage_1); %mean of projected area
mo=mode(OutputImage_1); %mode of projected area
med=median(OutputImage_1); %median of projected area
toc;

```

## APPENDIX II

---

### ❖ Python code for training encoder-decoder deep learning network.

```
'''Import the libraries'''
import os
import cv2
from keras.layers.core import *
from keras.layers import
Input,Dense,Flatten,Dropout,merge,Reshape,Conv2D,MaxPooling2D,UpSampling2
D,Conv2DTranspose,ZeroPadding2D, Add
from keras.layers.normalization import BatchNormalization
from keras.models import Model,Sequential,load_model
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adadelta, RMSprop,SGD,Adam
from keras import regularizers
from keras import backend as K
import numpy as np
import scipy
import numpy.random as rng
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
#from skimage.transform import resize
#from skimage.io import imsave
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

'''Set Keras image format '''
K.set_image_data_format('channels_last')

'''Define the model'''
##### Hourglass
#####

def Encoder(input_img):

    Econv1_1 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"block1_conv1")(input_img)
    Econv1_1 = BatchNormalization()(Econv1_1)
    Econv1_2 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"block1_conv2")(Econv1_1)
    Econv1_2 = BatchNormalization()(Econv1_2)
    pool1 = MaxPooling2D(pool_size=(2, 2),strides=(2,2),padding='same', name
= "block1_pool1")(Econv1_2)

    Econv2_1 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"block2_conv1")(pool1)
```

```

        Econv2_1 = BatchNormalization()(Econv2_1)
        Econv2_2 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"block2_conv2")(Econv2_1)
        Econv2_2 = BatchNormalization()(Econv2_2)
        pool2= MaxPooling2D(pool_size=(2, 2),strides=(2,2), padding='same', name
= "block2_pool1")(Econv2_2)

        Econv3_1 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"block3_conv1")(pool2)
        Econv3_1 = BatchNormalization()(Econv3_1)
        Econv3_2 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"block3_conv2")(Econv3_1)
        Econv3_2 = BatchNormalization()(Econv3_2)
        pool3 = MaxPooling2D(pool_size=(2, 2),strides=(2,2), padding='same', name
= "block3_pool1")(Econv3_2)

        encoded = Model(input = input_img, output = [pool3, Econv1_2, Econv2_2,
Econv3_2] )

        return encoded
##### Bottleneck
#####
#
##
def neck(input_layer):

        Nconv = Conv2D(256, (3,3),padding = "same", name = "neck1"
)(input_layer)
        Nconv = BatchNormalization()(Nconv)
        Nconv = Conv2D(128, (3,3),padding = "same", name = "neck2" )(Nconv)
        Nconv = BatchNormalization()(Nconv)

        neck_model = Model(input_layer, Nconv)
        return neck_model
#
##### Decoder
#####

def Decoder(inp ):

        up1 = Conv2DTranspose(128,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_1")(inp[0])
        up1 = BatchNormalization()(up1)
        up1 = merge([up1, inp[3]], mode='concat', concat_axis=3, name =
"merge_1")
        Upconv1_1 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"Upconv1_1")(up1)
        Upconv1_1 = BatchNormalization()(Upconv1_1)
        Upconv1_2 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"Upconv1_2")(Upconv1_1)

```

```

Upconv1_2 = BatchNormalization()(Upconv1_2)

up2 = Conv2DTranspose(64,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_2")(Upconv1_2)
up2 = BatchNormalization()(up2)
up2 = merge([up2, inp[2]], mode='concat', concat_axis=3, name =
"merge_2")
Upconv2_1 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"Upconv2_1")(up2)
Upconv2_1 = BatchNormalization()(Upconv2_1)
Upconv2_2 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"Upconv2_2")(Upconv2_1)
Upconv2_2 = BatchNormalization()(Upconv2_2)

up3 = Conv2DTranspose(16,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_3")(Upconv2_2)
up3 = BatchNormalization()(up3)
up3 = merge([up3, inp[1]], mode='concat', concat_axis=3, name =
"merge_3")
Upconv3_1 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"Upconv3_1")(up3)
Upconv3_1 = BatchNormalization()(Upconv3_1)
Upconv3_2 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"Upconv3_2")(Upconv3_1)
Upconv3_2 = BatchNormalization()(Upconv3_2)

decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same', name =
"Ouput_layer")(Upconv3_2)
convnet = Model(input = inp, output = decoded)
return convnet

#####

#####

#####"Initialise the
model."#####

x_shape = 256
y_shape = 256
channels = 3
i_s = 256
input_img = Input(shape = (x_shape, y_shape,channels))

#Encoder
encoded = Encoder(input_img)      #return encoded representation with
intermediate layer Pool3(encoded), Econv1_3, Econv2_3,Econv3_3

#Decoder
HG_ = Input(shape = (x_shape/(2**3),y_shape/(2**3),128))
conv1_1 = Input(shape = (x_shape,y_shape,16))

```

```

conv2_1 = Input(shape = (x_shape/(2**1),y_shape/(2**1),64))
conv3_1 = Input(shape = (x_shape/(2**2),y_shape/(2**2),128))
decoded = Decoder( [HG_, conv1_1, conv2_1, conv3_1])

#BottleNeck
Neck_input = Input(shape = (x_shape/(2**3), y_shape/(2**3),128))
neck = neck(Neck_input)

#Combined
output_img = decoded([neck(encoded(input_img)[0]), encoded(input_img)[1],
encoded(input_img)[2], encoded(input_img)[3]])
model= Model(input = input_img, output = output_img )
model.summary()
model.compile(optimizer = Adam(0.0005), loss='binary_crossentropy', metrics =
["accuracy"])
model.save_weights('Model_exp/UNet/Stats/UNet.h5')

#####
#####

name = os.listdir("/media/biometric/Data21/W/Autoencoder/Original_Data_120")
input_images = []
output_images = []

print("loading_images")
count = 0
for i in name :
    img_x =
cv2.imread("/media/biometric/Data21/W/Autoencoder/Original_Data_120/"+i)
    img_x = cv2.resize(img_x, (256,256))
    input_images.append(img_x)
    img_y =
cv2.imread("/media/biometric/Data21/W/Autoencoder/Mask_data_120/"+i.split(".")[
0] + "_mask" + ".png", 0)
    img_y = cv2.resize(img_y, (256,256))
    img_y = img_y[:, :, np.newaxis]
    output_images.append(img_y)
'''

print("converting to numpy arrays")
#input_images = np.asarray(input_images, np.float32) /255
#output_images = np.asarray(output_images, np.float32)/255
'''

print input_images[0].shape
print("Data_splitting..")
X_train,X_test,Y_train,Y_test=train_test_split(input_images,output_images,test_size
=0.3)
del input_images
del output_images

```

```

X_train = np.asarray(X_train, np.float16)/255
print("Done")
X_test = np.asarray(X_test, np.float16)/255
print("Done")
Y_train = np.asarray(Y_train, np.float16)/255
print("Done")
Y_test = np.asarray(Y_test, np.float16)/255
print("Done")
saveModel = "Model_exp/UNet/Stats/UNet.h5"
#numEpochs = 100
batch_size = 8
num_batches = int(len(X_train)/batch_size)
print "Number of batches: %d\n" % num_batches
saveDir = 'Model_exp/UNet/Stats/'
loss=[]
val_loss=[]
acc=[]
val_acc=[]
epoch=0;
best_loss=1000
r_c=0

while epoch <1001 :

    history=model.fit(X_train, Y_train, batch_size=batch_size, epochs=1,
validation_data=(X_test,Y_test), shuffle=True, verbose=1)
    #print float(history.history['loss'][0])

    #print 'INSIDE LOOP'

    model.save_weights(saveModel)

    epoch=epoch+1
    print "EPOCH NO. : "+str(epoch)+"\n"
    loss.append(float(history.history['loss'][0]))
    val_loss.append(float(history.history['val_loss'][0]))
    acc.append(float(history.history['acc'][0]))
    val_acc.append(float(history.history['val_acc'][0]))
    loss_arr=np.asarray(loss)
    e=range(epoch)
    plt.plot(e,loss_arr)
    plt.xlabel('Number of Epochs')
    plt.ylabel('Training Loss')
    plt.savefig('Model_exp/UNet/Stats/Plot'+str(epoch)+''.png')
    plt.close()
    loss1=np.asarray(loss)
    val_loss1=np.asarray(val_loss)
    acc1=np.asarray(acc)
    val_acc1=np.asarray(val_acc)

```

```

np.savetxt('Model_exp/UNet/Stats/Loss.txt',loss1)
np.savetxt('Model_exp/UNet/Stats/Val_Loss.txt',val_loss1)
np.savetxt('Model_exp/UNet/Stats/Acc.txt',acc1)
np.savetxt('Model_exp/UNet/Stats/Val_Acc.txt',val_acc1)

s=rng.randint(30)
x_test=X_test[s,:,:,:]
y_test=Y_test[s,:,:,:]
print(y_test.shape)
a = np.zeros([i_s, i_s,3])
a[:, :,0] = y_test[:, :,0]
a[:, :,1] = a[:, :,0]
a[:, :,2] = a[:, :,0]

#X_test,y_test = shuffle(X_test,y_test)
x_test=x_test.reshape(1,i_s, i_s,3)
y_test=a.reshape((1,i_s, i_s,3))
decoded_imgs = np.zeros([1,i_s, i_s,3])
print(model.predict(x_test)[0].shape)

decoded_imgs[:, :, :,0] = model.predict(x_test)[0][:, :,0]
decoded_imgs[:, :, :,1] = decoded_imgs[:, :, :,0]
decoded_imgs[:, :, :,2] = decoded_imgs[:, :, :,0]
temp = np.zeros([i_s, i_s*3,3])

temp[:, :i_s, :] = x_test[0, :, :, :]
temp[:, i_s:i_s*2, :] = decoded_imgs[0, :, :, :]
temp[:, i_s*2:, :] = y_test[0, :, :, :]

temp = temp*255
scipy.misc.imsave('Model_exp/UNet/' + str(epoch+1) + ".jpg", temp)

print("training Done.")
❖ Python code for testing encoder-decoder deep learning network.

```

```

"""Import the libraries"""
import os
import cv2
import numpy.ma as ma
from keras.layers.core import *
from keras.layers import
Input,Dense,Flatten,Dropout,merge,Reshape,Conv2D,MaxPooling2D,UpSampling2
D,Conv2DTranspose,ZeroPadding2D, Add
from keras.layers.normalization import BatchNormalization
from keras.models import Model,Sequential,load_model
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adadelta, RMSprop,SGD,Adam
from keras import regularizers
from keras import backend as K
import numpy as np

```

```

import scipy
import numpy.random as rng
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
#from skimage.transform import resize
#from skimage.io import imsave
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import tensorflow as tf

'''Set Keras image format '''
K.set_image_data_format('channels_last')
'''Define the model'''
##### Hourglass
#####

def Encoder(input_img):

    Econv1_1 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"block1_conv1")(input_img)
    Econv1_1 = BatchNormalization()(Econv1_1)
    Econv1_2 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"block1_conv2")(Econv1_1)
    Econv1_2 = BatchNormalization()(Econv1_2)
    pool1 = MaxPooling2D(pool_size=(2, 2),strides=(2,2),padding='same', name
= "block1_pool1")(Econv1_2)

    Econv2_1 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"block2_conv1")(pool1)
    Econv2_1 = BatchNormalization()(Econv2_1)
    Econv2_2 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"block2_conv2")(Econv2_1)
    Econv2_2 = BatchNormalization()(Econv2_2)
    pool2= MaxPooling2D(pool_size=(2, 2),strides=(2,2), padding='same', name
= "block2_pool1")(Econv2_2)

    Econv3_1 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"block3_conv1")(pool2)
    Econv3_1 = BatchNormalization()(Econv3_1)
    Econv3_2 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"block3_conv2")(Econv3_1)
    Econv3_2 = BatchNormalization()(Econv3_2)
    pool3 = MaxPooling2D(pool_size=(2, 2),strides=(2,2), padding='same', name
= "block3_pool1")(Econv3_2)

    encoded = Model(input = input_img, output = [pool3, Econv1_2, Econv2_2,
Econv3_2] )

    return encoded

```

```

##### Bottleneck
#####
#
##
def neck(input_layer):

    Nconv = Conv2D(256, (3,3),padding = "same", name = "neck1"
)(input_layer)
    Nconv = BatchNormalization()(Nconv)
    Nconv = Conv2D(128, (3,3),padding = "same", name = "neck2" )(Nconv)
    Nconv = BatchNormalization()(Nconv)

    neck_model = Model(input_layer, Nconv)
    return neck_model
#
##### Decoder
#####

def Decoder(inp ):

    up1 = Conv2DTranspose(128,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_1")(inp[0])
    up1 = BatchNormalization()(up1)
    up1 = merge([up1, inp[3]], mode='concat', concat_axis=3, name =
"merge_1")
    Upconv1_1 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"Upconv1_1")(up1)
    Upconv1_1 = BatchNormalization()(Upconv1_1)
    Upconv1_2 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"Upconv1_2")(Upconv1_1)
    Upconv1_2 = BatchNormalization()(Upconv1_2)

    up2 = Conv2DTranspose(64,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_2")(Upconv1_2)
    up2 = BatchNormalization()(up2)
    up2 = merge([up2, inp[2]], mode='concat', concat_axis=3, name =
"merge_2")
    Upconv2_1 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"Upconv2_1")(up2)
    Upconv2_1 = BatchNormalization()(Upconv2_1)
    Upconv2_2 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"Upconv2_2")(Upconv2_1)
    Upconv2_2 = BatchNormalization()(Upconv2_2)

    up3 = Conv2DTranspose(16,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_3")(Upconv2_2)
    up3 = BatchNormalization()(up3)
    up3 = merge([up3, inp[1]], mode='concat', concat_axis=3, name =
"merge_3")

```

```

        Upconv3_1 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"Upconv3_1")(up3)
        Upconv3_1 = BatchNormalization()(Upconv3_1)
        Upconv3_2 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"Upconv3_2")(Upconv3_1)
        Upconv3_2 = BatchNormalization()(Upconv3_2)

        decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same', name =
"Ouput_layer")(Upconv3_2)
        convnet = Model(input = inp, output = decoded)
        return convnet

#####
#####

x_shape = 256
y_shape = 256
channels = 3
i_s = 256
input_img = Input(shape = (x_shape, y_shape, channels))

#Encoder
encoded = Encoder(input_img)      #return encoded representation with
intermediate layer Pool3(encoded), Econv1_3, Econv2_3, Econv3_3

#Decoder
HG_ = Input(shape = (x_shape/(2**3), y_shape/(2**3), 128))
conv1_1 = Input(shape = (x_shape, y_shape, 16))
conv2_1 = Input(shape = (x_shape/(2**1), y_shape/(2**1), 64))
conv3_1 = Input(shape = (x_shape/(2**2), y_shape/(2**2), 128))
decoded = Decoder( [HG_, conv1_1, conv2_1, conv3_1])

#BottleNeck
Neck_input = Input(shape = (x_shape/(2**3), y_shape/(2**3), 128))
neck = neck(Neck_input)

#Combined
output_img = decoded([neck(encoded(input_img)[0]), encoded(input_img)[1],
encoded(input_img)[2], encoded(input_img)[3]])
model= Model(input = input_img, output = output_img )
model.summary()
model.compile(optimizer = Adam(0.0005), loss='binary_crossentropy', metrics =
["accuracy"])
model.load_weights('/media/biometric/Data21/W/Autoencoder/Model_exp/UNet/Sta
ts/UNet.h5')

#####
#####

```

```
#####Testing#####
#####
name = os.listdir("/media/biometric/Data21/W/Autoencoder/Test_Data/Original")
input_images = []
output_images = []

print("loading_images")
count = 0
for i in name :
    img_x =
cv2.imread("/media/biometric/Data21/W/Autoencoder/Test_Data/Original/"+i)
    img_x = cv2.resize(img_x, (256,256)) /255.0
    img_x = img_x.reshape(1,256,256,3)
    img_x = img_x.reshape(1,256,256,3)
    pred = model.predict(img_x)[0][:,0]
    pred = pred.reshape(256,256,1)
    cv2.imwrite("/media/biometric/Data21/W/Autoencoder/Test_Result/" + i,
pred *255)
```

#### ❖ Performance evaluation of the developed encoder-decoder model

```
'''Import the libraries'''
import os
import cv2
import numpy.ma as ma
from keras.layers.core import *
from keras.layers import
Input,Dense,Flatten,Dropout,merge,Reshape,Conv2D,MaxPooling2D,UpSampling2
D,Conv2DTranspose,ZeroPadding2D, Add
from keras.layers.normalization import BatchNormalization
from keras.models import Model,Sequential,load_model
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adadelta, RMSprop,SGD,Adam
from keras import regularizers
from keras import backend as K
import numpy as np
import scipy
import numpy.random as rng
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
#from skimage.transform import resize
#from skimage.io import imsave
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import tensorflow as tf

'''Set Keras image format '''
K.set_image_data_format('channels_last')
'''Define the model'''
```

```
##### Hourglass
#####
```

```
def Encoder(input_img):
```

```
    Econv1_1 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"block1_conv1")(input_img)
    Econv1_1 = BatchNormalization()(Econv1_1)
    Econv1_2 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"block1_conv2")(Econv1_1)
    Econv1_2 = BatchNormalization()(Econv1_2)
    pool1 = MaxPooling2D(pool_size=(2, 2),strides=(2,2),padding='same', name
= "block1_pool1")(Econv1_2)
```

```
    Econv2_1 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"block2_conv1")(pool1)
    Econv2_1 = BatchNormalization()(Econv2_1)
    Econv2_2 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"block2_conv2")(Econv2_1)
    Econv2_2 = BatchNormalization()(Econv2_2)
    pool2= MaxPooling2D(pool_size=(2, 2),strides=(2,2), padding='same', name
= "block2_pool1")(Econv2_2)
```

```
    Econv3_1 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"block3_conv1")(pool2)
    Econv3_1 = BatchNormalization()(Econv3_1)
    Econv3_2 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"block3_conv2")(Econv3_1)
    Econv3_2 = BatchNormalization()(Econv3_2)
    pool3 = MaxPooling2D(pool_size=(2, 2),strides=(2,2), padding='same', name
= "block3_pool1")(Econv3_2)
```

```
    encoded = Model(input = input_img, output = [pool3, Econv1_2, Econv2_2,
Econv3_2] )
```

```
    return encoded
```

```
##### Bottleneck
#####
```

```
#
##
```

```
def neck(input_layer):
```

```
    Nconv = Conv2D(256, (3,3),padding = "same", name = "neck1"
)(input_layer)
    Nconv = BatchNormalization()(Nconv)
    Nconv = Conv2D(128, (3,3),padding = "same", name = "neck2" )(Nconv)
    Nconv = BatchNormalization()(Nconv)
```

```
    neck_model = Model(input_layer, Nconv)
    return neck_model
```

```

#
##### Decoder
#####

def Decoder(inp ):

    up1 = Conv2DTranspose(128,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_1")(inp[0])
    up1 = BatchNormalization()(up1)
    up1 = merge([up1, inp[3]], mode='concat', concat_axis=3, name =
"merge_1")
    Upconv1_1 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"Upconv1_1")(up1)
    Upconv1_1 = BatchNormalization()(Upconv1_1)
    Upconv1_2 = Conv2D(128, (3, 3), activation='relu', padding='same', name =
"Upconv1_2")(Upconv1_1)
    Upconv1_2 = BatchNormalization()(Upconv1_2)

    up2 = Conv2DTranspose(64,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_2")(Upconv1_2)
    up2 = BatchNormalization()(up2)
    up2 = merge([up2, inp[2]], mode='concat', concat_axis=3, name =
"merge_2")
    Upconv2_1 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"Upconv2_1")(up2)
    Upconv2_1 = BatchNormalization()(Upconv2_1)
    Upconv2_2 = Conv2D(64, (3, 3), activation='relu', padding='same', name =
"Upconv2_2")(Upconv2_1)
    Upconv2_2 = BatchNormalization()(Upconv2_2)

    up3 = Conv2DTranspose(16,(3,3),strides = (2,2), activation = 'relu', padding
= 'same', name = "upsample_3")(Upconv2_2)
    up3 = BatchNormalization()(up3)
    up3 = merge([up3, inp[1]], mode='concat', concat_axis=3, name =
"merge_3")
    Upconv3_1 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"Upconv3_1")(up3)
    Upconv3_1 = BatchNormalization()(Upconv3_1)
    Upconv3_2 = Conv2D(16, (3, 3), activation='relu', padding='same', name =
"Upconv3_2")(Upconv3_1)
    Upconv3_2 = BatchNormalization()(Upconv3_2)

    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same', name =
"Ouput_layer")(Upconv3_2)
    convnet = Model(input = inp, output = decoded)
    return convnet

#####
#####

```

```

x_shape = 256
y_shape = 256
channels = 3
i_s = 256
input_img = Input(shape = (x_shape, y_shape, channels))

#Encoder
encoded = Encoder(input_img)      #return encoded representation with
intermediate layer Pool3(encoded), Econv1_3, Econv2_3, Econv3_3

#Decoder
HG_ = Input(shape = (x_shape/(2**3), y_shape/(2**3), 128))
conv1_1 = Input(shape = (x_shape, y_shape, 16))
conv2_1 = Input(shape = (x_shape/(2**1), y_shape/(2**1), 64))
conv3_1 = Input(shape = (x_shape/(2**2), y_shape/(2**2), 128))
decoded = Decoder( [HG_, conv1_1, conv2_1, conv3_1])

#BottleNeck
Neck_input = Input(shape = (x_shape/(2**3), y_shape/(2**3), 128))
neck = neck(Neck_input)

#Combined
output_img = decoded([neck(encoded(input_img)[0]), encoded(input_img)[1],
encoded(input_img)[2], encoded(input_img)[3]])
model= Model(input = input_img, output = output_img )
model.summary()
model.compile(optimizer = Adam(0.0005), loss='binary_crossentropy', metrics =
["accuracy"])
model.load_weights('/media/biometric/Data21/W/Autoencoder/Model_exp/UNet/Stats/UNet.h5')

#####
#####
#####Testing#####
#####

'''
name =
os.listdir("/media/biometric/Data21/W/Autoencoder/New_Data/Test/Original")
input_images = []
output_images = []

print("loading_images")
count = 0
for i in name :
    img_x =
cv2.imread("/media/biometric/Data21/W/Autoencoder/New_Data/Test/Original/"+i)

    img_x = cv2.resize(img_x, (256,256)) /255.0

```

```

img_x = img_x.reshape(1,256,256,3)
img_x = img_x.reshape(1,256,256,3)
pred = model.predict(img_x)[0][:,:,0]
pred = pred.reshape(256,256,1)
cv2.imwrite("/media/biometric/Data21/W/Autoencoder/New_Data/Test/Test
_Result/" + i, pred *255)
'''
#####
#####
#####Testing#####
#####

name = os.listdir("/media/biometric/Data21/Iris_All_Dataset/Interval_all_image")
input_images = []
gt_images = []
pred_images = []

print("loading_images")
count = 0
for i in name :
    if(os.path.exists("/media/biometric/Data21/Iris_All_Dataset/Interval_all_image/" +
i) and
os.path.exists("/media/biometric/Data21/Ranjeet/NI2SEGNET/hard_train/Hard_cases/
NEW/Test_on_New_Contour_Data/Interval_V3_Mask/" + i.split(".")[0] + '.png')):

        img_x =
cv2.imread("/media/biometric/Data21/Iris_All_Dataset/Interval_all_image/" + i, 0)

        img_x = cv2.resize(img_x, (256,256))
        img_x = img_x[:,:,:np.newaxis]
        input_images.append(img_x)
        img_y =
cv2.imread("/media/biometric/Data21/Ranjeet/NI2SEGNET/hard_train/Hard_cases/
NEW/Test_on_New_Contour_Data/Interval_V3_Mask/" + i.split(".")[0] + '.png', 0)
        img_y = cv2.resize(img_y, (256,256))
        img_y = img_y[:,:,:np.newaxis]
        gt_images.append(img_y)

        count+=1
        print count

#print (input_images[0].shape)
print("Data_conversion..")
input_images = np.asarray(input_images, np.float32)/255
gt_images = np.asarray(gt_images, np.float32)/255

#predicting the segmentation(iris)
pred_images = model.predict(input_images)
cv2.imwrite("1.jpg", gt_images[40]*255)
cv2.imwrite("2.jpg",pred_images[40]*255)

```

```

for i in range(len(pred_images)):
    temp = np.zeros([256,256*2])
    temp[:,256] = input_images[i,:,:,0]+gt_images[i,:,:,0]
    temp[:,256:] = input_images[i,:,:,0]+pred_images[i,:,:,0]
    cv2.imwrite("Results_Interval_V3_26_Jan/predicted/"+str(i)+".jpg",
temp*255)

```

```

def pixel_count(x):
    y = 0
    for j in range(256):
        for k in range(256):
            if x[j][k] >=0.5:
                y+=1
    return y

```

```

XOR=[]
f = open("Results_Interval_V3_26_Jan/E1_n.txt", "w+")
h = open("Results_Interval_V3_26_Jan/E2_n.txt", "w+")
g = open("Results_Interval_V3_26_Jan/JI_n.txt", "w+")
pr = open("Results_Interval_V3_26_Jan/Precision_n.txt", "w+")
rc = open("Results_Interval_V3_26_Jan/Recall_n.txt", "w+")
F1_m = open("Results_Interval_V3_26_Jan/F1_measure_n.txt", "w+")
Accuracy_t = open("Results_Interval_V3_26_Jan/Accuracy_n.txt", "w+")
Iris_JI_File = open("Results_Interval_V3_26_Jan/Iris_JI.txt", "w+")
input_images = input_images
gt_images = gt_images
pred_images = pred_images

```

```

E1 = 0
E2 = 0
A1 = 0
I11 = 0
NI11 = 0
PR_avg = 0
RC_avg = 0
F_1_avg = 0
Accuracy_avg = 0
Iris_JI = 0
for i in range(len(gt_images)) :
    xor =0
    uni = 0
    inters = 0
    gt_images[i,:,:,0] = (np.around(gt_images[i,:,:,0])).astype(int)
    pred_images[i,:,:,0] = (np.around(pred_images[i,:,:,0])).astype(int)
    print ("error_rate_ : "+str(i+1))
#classification error rate (E1)

```

```

xor = cv2.bitwise_xor(gt_images[i,:,:0],pred_images[i,:,:0])
Ei = float(np.sum(xor)) / float(256*256)
f.write(str(Ei)+"\n")
E1 += Ei
print("E1 : "+ str(Ei))
print(gt_images[i,:,:0])
print(pred_images[i,:,:0])
#classification error rate (E2)
FP = float(np.sum(cv2.bitwise_xor(pred_images[i,:,:0] ,
cv2.bitwise_and(gt_images[i,:,:0],pred_images[i,:,:0]))))
TN = float((256*256) - float(np.sum(gt_images[i,:,:0])))
FPR = float(FP) / float(FP+TN)

FN =float(np.sum(cv2.bitwise_xor(gt_images[i,:,:0],
cv2.bitwise_and(gt_images[i,:,:0],pred_images[i,:,:0]))))
TP = float(np.sum(gt_images[i,:,:0]))
FNR = float(FN) / float((FN+TP))

Ej = 0.5*FPR+0.5*FNR
h.write(str(Ej)+"\n")
E2 += Ej
print("E2 : "+str(Ej) )

# Jaccard Index (JI)
#uni = cv2.bitwise_or(gt_images[i,:,:0],pred_images[i,:,:0])
C11 = np.sum(cv2.bitwise_and(gt_images[i,:,:0],pred_images[i,:,:0]))
print('C11:' + str(C11))
G1 = np.sum(gt_images[i,:,:0])
print('G1:' + str(G1))
P1 = np.sum(pred_images[i,:,:0])
print('P1:' + str(P1))
Iris = C11/(G1 + P1 - C11)
Iris_JI = Iris_JI + Iris
print('Iris:' + str(Iris))
G_N = np.where(gt_images[i,:,:0] > 0.5, 1, 0)
P_N = np.where(pred_images[i,:,:0] > 0.5, 1, 0)
C22 = np.sum(cv2.bitwise_and(G_N, P_N))
print("C22:" + str(C22))
G2 = 65536 - G1
P2 = 65536 - P1
Non_Iris = C22/(G2 + P2 - C22)
JI = (Iris + Non_Iris)/2
g.write(str(JI)+"\n")
A1 += JI
TP = np.sum(cv2.bitwise_and(gt_images[i,:,:0],pred_images[i,:,:0]))
TN = np.sum(cv2.bitwise_and(G_N, P_N))
print('TN:' + str(TN))
FP = np.sum(pred_images[i,:,:0]) - TP
FN = np.sum(gt_images[i,:,:0]) - TP
Precision = TP/(TP + FP)

```

```

if math.isnan(Precision):
    Precision=0
print('Precision:', Precision)
Recall = TP / (TP + FN)
print('Recall:', Recall)
F1_measure = 2*Precision*Recall/(Precision + Recall)
if math.isnan(F1_measure):
    F1_measure = 0
print('F1_measure:',F1_measure)
Accuracy = (TP + TN)/(TP + TN + FP + FN)
print('Accuracy:',Accuracy)
PR_avg = PR_avg + Precision
RC_avg = RC_avg + Recall
F_1_avg = F_1_avg + F1_measure
Accuracy_avg = Accuracy_avg + Accuracy
pr.write(str(Precision)+"\n")
rc.write(str(Recall)+"\n")
F1_m.write(str(F1_measure)+"\n")
Accuracy_t.write(str(Accuracy)+"\n")
Iris_JI_File.write(str(Iris)+"\n")
print(len(gt_images))
print("classification error rate (E1) : " + str(E1/len(gt_images)) )
f.write("classification error rate (E1) : " + str(E1/len(gt_images))+"\n")
print("classification error rate (E2=avg(FPR+FNR)) : " + str(E2/len(gt_images)) )
f.write("classification error rate (E2=avg(FPR+FNR)) : " +
str(E2/len(gt_images))+"\n")
print("Average_Jaccard_Index : " + str(A1/len(gt_images)))
print("Average_Precision : " + str(PR_avg /(len(gt_images)-1)))
print("Average_Recall : " + str(RC_avg /len(gt_images)))
print("Average_F_1_measure : " + str(F_1_avg /(len(gt_images) - 1)))
print("Average_Accuracy: " + str(Accuracy_avg /len(gt_images)))
print("Average_Jaccard_Index_for_Iris: " + str(Iris_JI/len(gt_images)))
print("Testing Done.")

```

### ❖ Flask we development module code

```

import os
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename

UPLOAD_FOLDER = '/home/lc/Desktop/Flask_new/static/my'
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

imagefolder=os.path.join('static','my')
outimagefolder=os.path.join('static','output')
spike_count=os.path.join('static','Count_output')
home_img=os.path.join('static','HOME_IMG')

```

```

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['imagefolder'] = imagefolder
app.config['outimagefolder'] = outimagefolder
app.config['spike_count'] = spike_count
app.config['home_img'] = home_img

@app.route('/', methods = ['POST', 'GET'])
def test():
    f=os.listdir('/home/lc/Desktop/Flask_new/static/my')
    k=os.listdir('/home/lc/Desktop/Flask_new/static/output')
    l=os.listdir('/home/lc/Desktop/Flask_new/static/Count_output')
    for i in f:
        os.remove('static/my/'+i)
    for j in k:
        os.remove('static/output/'+j)
    for m in l:
        os.remove('static/Count_output/' + m)
    return render_template("home.html",home_img=home_img)

@app.route('/hello1', methods=['POST'])
def hello():
    first_name = request.form['first_name']
    last_name = request.form['last_name']
    return 'Hello %s %s have fun learning python <br/> <a href="/">Back Home</a>'
    % (first_name, last_name)

@app.route('/uploader', methods = ['POST', 'GET'])
def upload_file():
    # if 'photo' in request.files:
    for f in request.files.getlist('file'):
        filename = secure_filename(f.filename)
        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        filename_img = os.path.join(imagefolder, filename)
        #return render_template("showimg.html", user_img=filename_img)
    return render_template("insert.html", user_img=filename_img)

@app.route('/showimg', methods=['POST', 'GET'])
def show():
    # if 'photo' in request.files:

    f=os.listdir(imagefolder)
    return render_template("showimg.html", user_img=f)

@app.route('/RunModel', methods=['POST', 'GET'])
def RunModel():
    # if 'photo' in request.files:
    os.system('python3 /home/lc/Desktop/Flask_new/Test_Wheat_Autoencoder.py')

```

```

    f_in = os.listdir(imagefolder)
    f_out = os.listdir(outimagefolder)
    return render_template("showoutputimg.html", out_img=f_out, in_img=f_in)

@app.route('/Count', methods=['POST', 'GET'])
def Count():
    # if 'photo' in request.files:
    #os.system('python3 imageJ_Test.py')
    os.system('python3 /home/lc/Desktop/Flask_new/spike_count.py')
    #os.system('javac -cp .:ij149v.jar Test.java')
    #os.system('java -cp .:ij149v.jar Test')
    f_in = os.listdir(imagefolder)
    f_out = os.listdir(outimagefolder)
    s_count=os.listdir(spike_count)
    path = '/home/lc/Desktop/Flask_new/static/Count_output/'+ s_count[0]
    days_file = open(path, 'r')
    lines = days_file.readlines()
    y = [0]
    z = []
    for x in lines:
        # print(x.split(",")[1])
        z.append(x.split(",")[1])

    for i in range(len(z)):
        t = int(z[i])
        y.append(t)

    max_count=max(y)
    return render_template("Count_page.html", out_img=f_out,
in_img=f_in,line=lines,max_count=max_count,spike_count=s_count[0])

if __name__ == '__main__':
    app.run()

```

### ❖ Spike counting code

```

import ij.IJ;
import ij.ImagePlus;
import ij.measure.ResultsTable;
import static ij.plugin.FFT.fileName;
import ij.plugin.PlugIn;
import ij.plugin.filter.Analyzer;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.*;

```

```

import java.util.Random;

public class spike_count_java_new implements PlugIn {

    public static void main(String[] args) {

        spike_count_java_new test = new spike_count_java_new();
        test.run("");

    }

    @Override
    public void run(String arg0) {

        try {
            // FileWriter fileWriter = new
FileWriter("C:\\Users\\Tanuj\\Desktop\\Book3.txt");
            Random rand = new Random();
            int n = rand.nextInt(50);

            PrintWriter pw = new PrintWriter(new
File("/home/lc/Desktop/Flask_new/static/Count_output/test_"+n+".txt"));

            //BufferedWriter bw = new BufferedWriter(fileWriter);
            //bw.write(content);

            String line = "\n";

            File file = new File("/home/lc/Desktop/Flask_new/static/output/");
            String[] fileList = file.list();

            for(String name:fileList){
                System.out.println(name);
            }

            for(String name:fileList){
                StringBuilder sb = new StringBuilder();

                ImagePlus imp =
IJ.openImage("/home/lc/Desktop/Flask_new/static/output/" + name);
                // FileSaver fileSaver = new FileSaver(imp);

                // System.setProperty("plugins.dir", "D:\\Tanuj\\SOFTWARES\\ImageJ2-
20160205\\ImageJ.app\\plugins");
                // IJ.run(imp, "canny.txt", "sigma=5 smoothing_factor=1");
                // fileSaver.saveAsJpeg("C:\\Users\\Tanuj\\Desktop\\0_7\\011_mask.txt");

```

```

        IJ.run(imp, "Convert to Mask", "");
        IJ.run(imp, "Analyze Particles...", " clear summarize stack");
    // IJ.renameResults("Results");
    // fileSaver.saveAsText("C:\\Users\\Tanuj\\Desktop\\0_7
Output\\011_mask.txt");

    ResultsTable rt = Analyzer.getResultsTable();
    int counter = rt.getCounter(); //number of results
    System.out.println("Object Count = " + counter + "\t");
    int col = 0;

    double total_area = 0.0;
    if (counter == 0) //no results, handle that error here
    {
        col = rt.getColumnIndex("Area");
    }
    int tmp_counter = 0;
    double tmp_value = 0.0;
    for (int row = 0; row < counter; row++) {
        double value = rt.getValueAsDouble(col, row); //all the Area values

        // System.out.println(value);
        if (value <= 5.0) {
            tmp_counter = tmp_counter + 1;
            tmp_value = tmp_value + value;
            // counter=counter-1;
            System.out.println(tmp_counter + "\t" + tmp_value);
            // System.out.println(tmp_value+"\t");
        }else{

            total_area = total_area + value;
            // System.out.println("\t"+total_area);
        }
    }
    System.out.println(name+"" + "\t Ear Count = " + (counter - tmp_counter)
+ "\t and total area = " + (total_area - tmp_value));
    System.out.println("#####");

    //fileWriter.append("\n");
    sb.append(name + "");
    sb.append(',');
    sb.append((counter - tmp_counter));
    sb.append(',');
    sb.append((total_area - tmp_value));
    sb.append("\r\n");

    // fileWriter.write("\n");
    // bw.close();
    //fileWriter.flush();

```

```
        // bw.newLine();

        pw.write(sb.toString());

        //pw.write(sb.toString());
    }
    pw.close();
} catch (IOException ex) {
    //Logger.getLogger(Test.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
```

## ANNEXURE I

---

### Actual Leaf Fresh Weight (LFW) of 104 dataset of Rice plants

Sample ID	Actual LFW
1	0.8547
2	0.986
3	1.5285
4	1.0472
5	1.2852
6	1.3781
7	2.1308
8	0.5684
9	0.8437
10	1.1199
11	1.4649
12	1.2826
13	1.14186
14	0.9365
15	1.03
16	2.0161
17	1.3291
18	1.5938
19	1.1628
20	0.9259
21	1.2055
22	0.7815
23	1.4402

24	0.9532
25	1.3678
26	1.0217
27	1.7201
28	1.513
29	1.1411
30	1.6886
31	1.1756
32	1.4062
33	0.8926
34	1.1614
35	0.648
36	1.259
37	1.5472
38	1.3694
39	1.1427
40	1.4896
41	1.0869
42	0.9999
43	1.4485
44	1.6742
45	1.3625
46	1.8072
47	1.1131
48	1.0112
49	0.9328
50	1.2973

51	1.0613
52	1.3586
53	0.8439
54	1.1995
55	0.902
56	0.8238
57	1.0153
58	1.7563
59	1.0304
60	1.9593
61	1.1409
62	1.8024
63	1.0957
64	1.1691
65	0.7167
66	1.1462
67	1.3058
68	1.0206
69	1.5849
70	0.3107
71	1.251
72	1.1756
73	1.0149
74	0.8508
75	1.8085
76	1.5708
77	1.3974

78	1.2991
79	1.35883
80	1.101
81	1.5229
82	1.2709
83	0.8471
84	1.2769
85	1.1751
86	1.1947
87	1.1342
88	1.5372
89	1.31
90	1.0147
91	0.8961
92	1.5938
93	1.6712
94	1.1343
95	0.7515
96	1.06
97	1.4586
98	1.3622
99	1.3258
100	0.6828
101	1.2629
102	0.8524
103	1.1232
104	0.8715

**Regression and ANN result of predicted LFW for Training data (85% data)**

<b>Sample number</b>	<b>Actual</b>	<b>Regression</b>	<b>ANN</b>
1	0.8547	1.194357411	1.090467
2	0.986	1.157517543	0.708862
3	1.5285	1.040767648	1.512766
4	1.0472	1.329587674	1.157463
5	1.2852	1.31778662	1.321762
6	1.3781	1.281891172	1.318168
7	2.1308	1.230841021	2.191886
8	0.5684	1.154673087	1.061525
9	0.8437	1.072166977	1.02361
10	1.1199	1.31742561	1.069163
11	1.4649	1.231542323	1.344454
12	1.2826	1.306094384	1.265604
13	1.14186	1.222134341	1.090421
14	0.9365	1.037058562	1.050514
15	1.03	1.350090375	1.032488
16	2.0161	1.210703962	1.707225
17	1.3291	1.237832178	1.186294
18	1.5938	1.351629717	1.563844
19	1.1628	1.345121227	1.182816
20	0.9259	1.39133077	1.117523
21	1.2055	1.289937842	1.221875
22	0.7815	1.163375781	0.755871
23	1.4402	1.20974974	1.507946
24	0.9532	1.16398516	1.045079

25	1.3678	1.162000449	1.15672
26	1.0217	1.23906742	1.136381
27	1.7201	1.314154293	1.733475
28	1.513	1.034540759	1.037258
29	1.1411	1.223908571	1.301314
30	1.6886	1.245485987	1.605727
31	1.1756	1.155936575	1.058107
32	1.4062	1.315272872	1.113947
33	0.8926	1.160226407	1.009922
34	1.1614	1.181205828	1.101409
35	0.648	1.120944452	1.188074
36	1.259	1.213823189	1.083776
37	1.5472	1.247229638	1.458849
38	1.3694	1.099242002	1.229204
39	1.1427	1.32430285	1.211172
40	1.4896	1.442789078	1.388392
41	1.0869	1.136153343	1.087626
42	0.9999	1.269812171	1.053476
43	1.4485	1.218449762	1.115781
44	1.6742	1.212202673	1.649836
45	1.3625	1.405744902	1.475658
46	1.8072	1.409424576	1.756091
47	1.1131	1.139430733	1.090061
48	1.0112	1.115552635	1.003306
49	0.9328	1.219513871	1.140082
50	1.2973	1.242276454	1.203397
51	1.0613	1.188951593	0.904732

52	1.3586	1.23395552	1.412873
53	0.8439	1.240021613	1.128721
54	1.1995	1.105878927	1.256914
55	0.902	1.174674316	0.900025
56	0.8238	1.231793823	1.109479
57	1.0153	1.348974013	1.000861
58	1.7563	1.316077404	1.626006
59	1.0304	1.052705879	1.069268
60	1.9593	1.367238849	1.87805
61	1.1409	1.074936989	1.146205
62	1.8024	1.319290168	1.81509
63	1.0957	1.234693579	1.18846
64	1.1691	1.285222578	1.214118
65	0.7167	1.127231334	0.830305
66	1.1462	1.186850882	0.964212
67	1.3058	1.215602735	1.223091
68	1.0206	1.212643107	1.098573
69	1.5849	1.143270845	1.552766
70	0.3107	1.264534295	0.313068
71	1.251	1.342620205	1.343419
72	1.1756	1.232796484	1.073729
73	1.0149	1.224417152	1.09809
74	0.8508	1.119884454	0.795994
75	1.8085	1.305469265	1.83776
76	1.5708	1.253538038	1.521117
77	1.3974	1.139111008	1.406595
78	1.2991	1.121819981	1.345173

79	1.35883	1.277461254	1.248263
80	1.101	1.382078359	1.068182
81	1.5229	1.310939948	1.421569
82	1.2709	1.226933971	1.388138
83	0.8471	1.190496214	1.154481
84	1.2769	1.363853325	1.29037
85	1.1751	1.324899436	1.173334
86	1.1947	1.318138314	1.108916
87	1.1342	1.179875328	1.129137
88	1.5372	1.219216198	1.378671
<b>RMSE</b>		<b>0.311902918</b>	<b>0.154417</b>
<b>MAPE</b>		<b>23.44</b>	<b>9.55</b>

**Regression and ANN result of predicted LFW for Testing data (15%)**

<b>Sample number</b>	<b>Actual</b>	<b>Regression</b>	<b>ANN</b>
1	1.31	1.462116983	1.121451
2	1.0147	1.273211328	0.967244
3	0.8961	1.283466754	1.062385
4	1.5938	1.337284528	1.703273
5	1.6712	1.043436518	1.469714
6	1.1343	1.283009075	1.290553
7	0.7515	1.447989148	1.044846
8	1.06	1.110433963	1.133104
9	1.4586	1.15528355	1.304589
10	1.3622	1.305172784	1.359343
11	1.3258	1.358812454	1.303432

12	0.6828	1.078149895	0.637565
13	1.2629	1.309835282	1.276677
14	0.8524	1.079990963	0.93024
15	1.1232	1.244293724	1.147654
16	0.8715	1.119716541	0.930143
<b>RMSE</b>		<b>0.315493959</b>	<b>0.130001</b>
<b>MAPE</b>		<b>24.97</b>	<b>9.65</b>

## ANNEXURE II

### Wheat variety data of 200 plants:

Plant ID	Genotype	Spike count
0	10_C_Choti Lerma_NUE_R1	14
1	10_C_Choti Lerma_NUE_R2	11
2	10_C_Choti Lerma_NUE_R3	15
3	10_D_Choti Lerma_NUE_R1	10
4	10_D_Choti Lerma_NUE_R2	8
5	10_D_Choti Lerma_NUE_R3	10
6	10_N_Choti Lerma_NUE_R1	14
7	10_N_Choti Lerma_NUE_R2	11
8	10_N_Choti Lerma_NUE_R3	9
9	11_C_HD2851_NUE_R1	9
10	11_C_HD2851_NUE_R2	4
11	11_C_HD2851_NUE_R3	13
12	11_D_HD2851_NUE_R1	10
13	11_D_HD2851_NUE_R2	10
14	11_D_HD2851_NUE_R3	8
15	11_N_HD2851_NUE_R1	13
16	11_N_HD2851_NUE_R2	14
17	11_N_HD2851_NUE_R3	16
18	12_C_HUW468_NUE_R1	12
19	12_C_HUW468_NUE_R2	8
20	12_C_HUW468_NUE_R3	12
21	12_D_HUW468_NUE_R1	9
22	12_D_HUW468_NUE_R2	11
23	12_D_HUW468_NUE_R3	8
24	12_N_HUW468_NUE_R1	11
25	12_N_HUW468_NUE_R2	10
26	12_N_HUW468_NUE_R3	11
27	13_C_HD2985_NUE_R1	5
28	13_C_HD2985_NUE_R2	8
29	13_C_HD2985_NUE_R3	16
30	13_D_HD2985_NUE_R1	6
31	13_D_HD2985_NUE_R2	7
32	13_D_HD2985_NUE_R3	10
33	13_N_HD2985_NUE_R1	9
34	13_N_HD2985_NUE_R2	8
35	13_N_HD2985_NUE_R3	7
36	14_C_HD3086_NUE_R1	18
37	14_C_HD3086_NUE_R2	12

38	14_C_HD3086_NUE_R3	12
39	14_D_HD3086_NUE_R1	12
40	14_D_HD3086_NUE_R2	11
41	14_D_HD3086_NUE_R3	12
42	14_N_HD3086_NUE_R1	9
43	14_N_HD3086_NUE_R2	9
44	14_N_HD3086_NUE_R3	12
45	15_C_HD2009_NUE_R1	13
46	15_C_HD2009_NUE_R2	12
47	15_C_HD2009_NUE_R3	14
48	15_D_HD2009_NUE_R1	10
49	15_D_HD2009_NUE_R2	9
50	15_D_HD2009_NUE_R3	5
51	15_N_HD2009_NUE_R1	12
52	15_N_HD2009_NUE_R2	13
53	15_N_HD2009_NUE_R3	11
54	16_C_Sytnetic 46_NUE_R1	22
55	16_C_Sytnetic 46_NUE_R2	7
56	16_C_Sytnetic 46_NUE_R3	28
57	16_D_Sytnetic 46_NUE_R1	14
58	16_D_Sytnetic 46_NUE_R2	14
59	16_D_Sytnetic 46_NUE_R3	12
60	16_N_Sytnetic 46_NUE_R1	11
61	16_N_Sytnetic 46_NUE_R2	14
62	16_N_Sytnetic 46_NUE_R3	14
63	17_C_HD2932_NUE_R1	6
64	17_C_HD2932_NUE_R2	6
65	17_C_HD2932_NUE_R3	10
66	17_D_HD2932_NUE_R1	8
67	17_D_HD2932_NUE_R2	5
68	17_D_HD2932_NUE_R3	8
69	17_N_HD2932_NUE_R1	12
70	17_N_HD2932_NUE_R2	12
71	17_N_HD2932_NUE_R3	10
72	18_C_GW322_NUE_R1	0
73	18_C_GW322_NUE_R2	15
74	18_C_GW322_NUE_R3	15
75	18_D_GW322_NUE_R1	9
76	18_D_GW322_NUE_R2	9
77	18_D_GW322_NUE_R3	12
78	18_N_GW322_NUE_R1	10
79	18_N_GW322_NUE_R2	12
80	18_N_GW322_NUE_R3	10

81	19_C_HD2987_NUE_R1	10
82	19_C_HD2987_NUE_R2	9
83	19_C_HD2987_NUE_R3	13
84	19_D_HD2987_NUE_R1	11
85	19_D_HD2987_NUE_R2	10
86	19_D_HD2987_NUE_R3	11
87	19_N_HD2987_NUE_R1	12
88	19_N_HD2987_NUE_R2	16
89	19_N_HD2987_NUE_R3	15
90	1_C_Lok1_NUE_R1	16
91	1_C_Lok1_NUE_R2	13
92	1_C_Lok1_NUE_R3	14
93	1_D_Lok1_NUE_R1	2
94	1_D_Lok1_NUE_R2	0
95	1_D_Lok1_NUE_R3	15
96	1_N_Lok1_NUE_R1	17
97	1_N_Lok1_NUE_R2	17
98	1_N_Lok1_NUE_R3	14
99	20_C_HD 2888_NUE_R1	14
100	20_C_HD 2888_NUE_R2	29
101	20_C_HD 2888_NUE_R3	27
102	20_D_HD 2888_NUE_R1	17
103	20_D_HD 2888_NUE_R2	21
104	20_D_HD 2888_NUE_R3	18
105	20_N_HD 2888_NUE_R1	19
106	20_N_HD 2888_NUE_R2	15
107	20_N_HD 2888_NUE_R3	18
108	21_C_HI 1500_NUE_R1	25
109	21_C_HI 1500_NUE_R2	20
110	21_C_HI 1500_NUE_R3	26
111	21_D_HI 1500_NUE_R1	27
112	21_D_HI 1500_NUE_R2	16
113	21_D_HI 1500_NUE_R3	18
114	21_N_HI 1500_NUE_R1	13
115	21_N_HI 1500_NUE_R2	6
116	21_N_HI 1500_NUE_R3	16
117	22_C_PBW550_NUE_R1	9
118	22_C_PBW550_NUE_R2	8
119	22_C_PBW550_NUE_R3	10
120	22_D_PBW550_NUE_R1	9
121	22_D_PBW550_NUE_R2	12
122	22_D_PBW550_NUE_R3	10
123	22_N_PBW550_NUE_R1	10

124	22_N_PBW550_NUE_R2	11
125	22_N_PBW550_NUE_R3	8
126	23_C_Kauz-AA-Kauz_NUE_R1	10
127	23_C_Kauz-AA-Kauz_NUE_R2	11
128	23_C_Kauz-AA-Kauz_NUE_R3	7
129	23_D_Kauz-AA-Kauz_NUE_R1	7
130	23_D_Kauz-AA-Kauz_NUE_R2	8
131	23_D_Kauz-AA-Kauz_NUE_R3	12
132	23_N_Kauz-AA-Kauz_NUE_R1	10
133	23_N_Kauz-AA-Kauz_NUE_R2	8
134	23_N_Kauz-AA-Kauz_NUE_R3	10
135	24_C_DharwadDry_NUE_R1	1
136	24_C_DharwadDry_NUE_R2	18
137	24_C_DharwadDry_NUE_R3	18
138	24_D_DharwadDry_NUE_R1	11
139	24_D_DharwadDry_NUE_R2	14
140	24_D_DharwadDry_NUE_R3	11
141	24_N_DharwadDry_NUE_R1	9
142	24_N_DharwadDry_NUE_R2	9
143	24_N_DharwadDry_NUE_R3	8
144	25_C_RAC 875_NUE_R1	14
145	25_C_RAC 875_NUE_R2	14
146	25_C_RAC 875_NUE_R3	17
147	25_D_RAC 875_NUE_R1	9
148	25_D_RAC 875_NUE_R2	17
149	25_D_RAC 875_NUE_R3	13
150	25_N_RAC 875_NUE_R1	14
151	25_N_RAC 875_NUE_R2	10
152	25_N_RAC 875_NUE_R3	12
153	26_C_BT Schumburk_NUE_R1	23
154	26_C_BT Schumburk_NUE_R2	32
155	26_C_BT Schumburk_NUE_R3	25
156	26_D_BT Schumburk_NUE_R1	19
157	26_D_BT Schumburk_NUE_R2	23
158	26_D_BT Schumburk_NUE_R3	19
159	26_N_BT Schumburk_NUE_R1	14
160	26_N_BT Schumburk_NUE_R2	17
161	26_N_BT Schumburk_NUE_R3	17
162	27_C_Kater-1_NUE_R1	19
163	27_C_Kater-1_NUE_R2	0
164	27_C_Kater-1_NUE_R3	11
165	27_D_Kater-1_NUE_R1	11
166	27_D_Kater-1_NUE_R2	9

167	27_D_Kater-1_NUE_R3	8
168	27_N_Kater-1_NUE_R1	9
169	27_N_Kater-1_NUE_R2	6
170	27_N_Kater-1_NUE_R3	9
171	28_C_HI1500_old_NUE_R1	0
172	28_C_HI1500_old_NUE_R2	8
173	28_C_HI1500_old_NUE_R3	8
174	28_D_HI1500_old_NUE_R1	8
175	28_D_HI1500_old_NUE_R2	7
176	28_D_HI1500_old_NUE_R3	8
177	28_N_HI1500_old_NUE_R1	10
178	28_N_HI1500_old_NUE_R2	8
179	28_N_HI1500_old_NUE_R3	9
180	29_C_DBW-43_NUE_R1	13
181	29_C_DBW-43_NUE_R2	11
182	29_C_DBW-43_NUE_R3	16
183	29_D_DBW-43_NUE_R1	8
184	29_D_DBW-43_NUE_R2	8
185	29_D_DBW-43_NUE_R3	13
186	29_N_DBW-43_NUE_R1	12
187	29_N_DBW-43_NUE_R2	12
188	29_N_DBW-43_NUE_R3	19
189	2_C_MP-4010_NUE_R1	3
190	2_C_MP-4010_NUE_R2	8
191	2_C_MP-4010_NUE_R3	7
192	2_D_MP-4010_NUE_R1	0
193	2_D_MP-4010_NUE_R2	7
194	2_D_MP-4010_NUE_R3	2
195	2_N_MP-4010_NUE_R1	9
196	2_N_MP-4010_NUE_R2	8
197	2_N_MP-4010_NUE_R3	9
198	30_C_NI5439_NUE_R1	9
199	30_C_NI5439_NUE_R2	8