IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received April 5, 2021, accepted May 5, 2021, date of publication May 17, 2021, date of current version June 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3080836

Web-SpikeSegNet: Deep Learning Framework for Recognition and Counting of Spikes From Visual Images of Wheat Plants

TANUJ MISRA[1,5], ALKA ARORA[1], SUDEEP MARWAHA[1], RANJEET RANJAN JHA[2], MRINMOY RAY[1], RAJNI JAIN[3], A. R. RAO[1], ELDHO VARGHESE[4], SHAILENDRA KUMAR[5], SUDHIR KUMAR[6], ADITYA NIGAM[2], RABI NARAYAN SAHOO[6], AND VISWANATHAN CHINNUSAMY[6]

[1]ICAR-Indian Agricultural Statistics Research Institute, New Delhi 110012, India
[2]School of Computing and Electrical Engineering (SCEE), IIT Mandi, Mandi 175005, India
[3]ICAR—National Institute of Agricultural Economics and Policy Research (NIAP), New Delhi 110012, India
[4]ICAR-Central Marine Fisheries Research Institute, Kochi 682018, India
[5]Department of Computer Science, Rani Lakshmi Bai Central Agricultural University, Jhansi 284003, India
[6]ICAR-Indian Agricultural Research Institute, Library Avenue, New Delhi 110012, India

Corresponding author: Alka Arora (Alka.Arora@icar.gov.in)

This work was supported in part by the National Agriculture Science Fund (NASF) through ICAR under Grant NASF/Phen-6005/2016-17, and in part by National Agricultural Higher Education Project-Centre of Advanced Agricultural Science and Technology (NAHEP CAAST) under Grant NAHEP/CAAST/2018/19/07. The work of Tanuj Misra was supported by the Fellowship from the Indian Council of Agricultural Research-Indian Agricultural Statistic Research Institute (ICAR-IASRI), New Delhi, India.

ABSTRACT Computer vision with deep learning is emerging as a significant approach for non-invasive and non-destructive plant phenotyping. Spikes are the reproductive organs of wheat plants. Detection and counting of spikes considered the grain-bearing organ have great importance in the phenomics study of large sets of germplasms. In the present study, we developed an online platform, ''Web-SpikeSegNet,'' based on a deep-learning framework for spike detection and counting from the wheat plant's visual images. The architecture of the Web-SpikeSegNet consists of 2 layers. First Layer, Client-Side Interface Layer, deals with end user's requests and corresponding responses management. In contrast, the second layer, Server Side Application Layer, consists of a spike detection and counting module. The backbone of the spike detection module comprises of deep encoder-decoder network with hourglass network for spike segmentation. The Spike counting module implements the ''Analyze Particle'' function of imageJ to count the number of spikes. For evaluating the performance of Web-SpikeSegNet, we acquired the wheat plant's visual images, and the satisfactory segmentation performances were obtained as Type I error 0.00159, Type II error 0.0586, Accuracy 99.65%, Precision 99.59% and F_1 score 99.65%. As spike detection and counting in wheat phenotyping are closely related to the yield, Web-SpikeSegNet is a significant step forward in the field of wheat phenotyping and will be very useful to the researchers and students working in the domain.

INDEX TERMS Computer vision, deep learning, deep encoder-decoder, hourglass, image analysis, spike detection and counting, Web-SpikeSegNet, wheat.

I. INTRODUCTION

Wheat is one of the major food crops grown yearly on 215 million hectares globally [Wheat in the world CGIAR: *https://wheat.org/wheat -in-the-world/*]. It supersedes maize and rice in terms of protein sources in low- and middle-income nations. Climate change and associated abiotic stresses are the key factors of yield loss in Wheat. Generic improvement in yield and climate resilience is critical for sustainable food security. One of the key aspects of genetic improvement is the determination of complex genome × environment × management interactions [1]. High-dimensional plant phenotyping is needed to bridge the genotype-phenotype gap in plant breeding and plant health monitoring in precision farming. Visual imaging is the most commonly used cost-effective method to quantitatively study plant growth, yield, and adaptation of biotic and abiotic stresses. Besides, it is strongly reasoned that the imminent trend in plant phenotyping will depend on imaging sensors' combined tools and machine learning [2]. Yield estimation

The associate editor coordinating the review of this manuscript and approving it for publication was Tallha Akram.

in Wheat has received significant attention from researchers. The number of spikes/ears determines the grain number per unit area and thus yield. Counting spikes of a large number of genotypes through traditional methods using naked-eye is a tedious and time-consuming job. Presently, non-destructive image analysis-based phenotyping is gaining momentum and proves as the less laborious and fast method. A cluster of research works available in the area of computer vision to detect and characterize spikes and spikelets in wheat plants [3]–[8]. High-resolution image dataset with significant quantity is a major constraint to develop the computer vision based approaches. In this context, Pound *et al.* [6] and David *et al.* [9] contributed ACID (Annotated Crop Image Dataset) and GWHD (Global Wheat Head Detection) dataset respectively. In computer vision, the problem of spike detection lies under the domain of pixel-wise segmentation of objects. Bi *et al.* [4], Qiongyan *et al.* [5] and Sadeghi-Tehran *et al.* [7] used manually defined color intensities and textures for spike segmentation. Pound *et al.* [6] and Hasan *et al.* [8] used Autoencoder [10] and Region-based Convolutional Neural Network (R-CNN) [10] deep-learning technique, respectively, to detect and characterize spikes with greater than 90 percent accuracy. Xiong *et al.* [11] proposed a deep-learning model "TasselNetV2" to characterize the maize tassels with around 91% accuracy. Sadeghi-Tehran *et al.* [12] developed a methodology using Simple Linear Iterative Clustering and Deep Convolutional Neural Networks for the spike quantification in the wheat plant. Recently, Misra *et al.* [3] developed a deep learning model known as SpikeSegNet, which was reported as an effective and robust approach for spike detection (accuracy: 99.91 percent) and counting (accuracy: 95 percent) from visual images irrespective of various illumination factors. In this paper, a web solution is presented as "Web-SpikeSegNet" for spike segmentation and counting from wheat plants' visual images for easy accessibility and quick reference. The developed web solution has a wide application in the plant phenomics domain and will be useful for researchers and students working in the field of wheat plant phenotyping. Web-SpikeSegNet is platform-independent and is readily accessible by at the URL: http://spikesegnet.iasri.res.in/.

II. IMPLEMENTATION

Web_SpikeSegNet is developed based on the approach given by Misra *et al.* [3]. The approach is based on the convolutional encoder-decoder deep-learning technique for pixel-wise segmentation of spikes from the wheat plant's visual images. The architecture of the network was inspired by UNet [13], SegNet [14], and PixISegNet [15], which are popularly used in various sectors for pixel-wise segmentation of objects. SpikeSegNet consists of two modules *viz.,* Local Patch extraction Network (LPNet) and Global Mask Refinement Network (GMRNet), in sequential order. The details of the approach are given in [3]. Input images were divided into patches before entering into the LPNet module to facilitate local features' learning more effectively than the whole input image. LPNet was used in extracting and understanding the contextual and local features at the patch level. Output images of the LPNet are further refined at GMRNet to better segment the spikes, as given in Figure 1. SpikeSegNet network was trained using visual images of the wheat plant and its corresponding ground-truth segmented mask images with class labels (*i.e.,* spike regions of the plant image). Details of the dataset preparation for training the network were given in [3]. SpikeSegNet provides significant segmentation performance at pixel-level in spike detection and counting and is also proved as a robust approach when tested for different illumination levels that may occur in the field conditions.

A. ARCHITECTURE OF THE PROPOSED SOFTWARE – "Web-SpikeSegNet"

Web-SpikeSegNet is web-based software for the detection and counting of spikes from visual images of the wheat plant. It is developed and implemented on the Linux operating system with 32 GB RAM and NVIDIA GeForce GTX 1080 Ti graphics card (with 11 GB). PyCharm version 5.0 integrative development environment [*https://www.jetbrains.com/*] was used to develop the software. The software architecture consists of two layers: Client-Side Interface Layer (CSIL) and Server Side Application Layer (SSAL). The architecture of Web-SpikeSegNet is given in Fig. 2. End-users (especially the plant physiologist) will interact with the Web-SpikeSegNet available at *http://spikesegnet.iasri.res.in/* through CSIL using internet. CSIL deals with the end-users requests and its corresponding responses management and implemented using HyperText Markup Language (HTML) [16], Cascading Style Sheets (CSS) [17], Flask [18], and JavaScript [19] technologies. HTML, CSS, and Flask were used to design the front-end view of the webpages, and JavaScript was used for the client-side validation. End-users will upload wheat image in the software through CSIL and then it will be forwarded to the SSAL for spike detection and counting. SSAL consists of two modules: spike detection and spike counting module. SpikeSegNet deep learning model will be applied on the input image for the spike segmentation in the Spike Detection module, and it will be forwarded to the spike counting module for counting the segmented spikes. After completion of the process, the segmented spikes along with spike count will be shown in the end-users window through CSIL. Spike detection module was developed using python libraries such as Tensorflow [20], Keras [21], Numpy [22], Scipy [23], Matplotlib [24] and OpenCV [25] for constructing and implementing the deep learning model. Convolutional encoder network [10] (Encoder_SpikeSegNet), decoder network [10] (Decoder_SpikeSegNet), and bottleneck network ([10], [15]) using stacked hourglasses (Bottleneck_SpikeSegNet) are the backbone of LPNet, GMRNet and correspondingly the SpikeSegNet. The number of encoders, decoders, and stacked hourglasses was estimated empirically, as given in [3], to produce the best results by considering the optimum performances. Encoder_SpikeSegNet

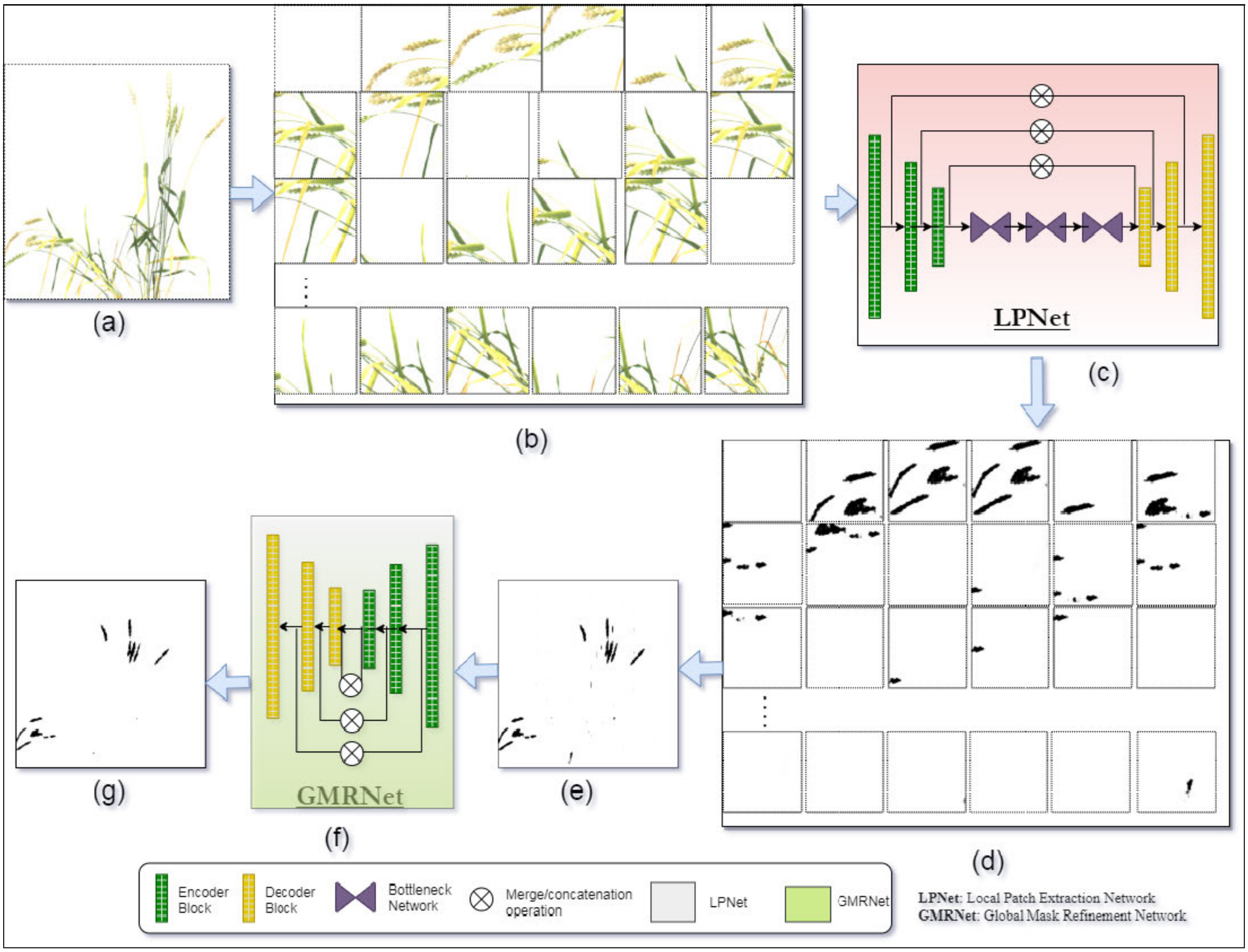


FIGURE 1. Flow diagram of SpikeSegNet: Here, input is visual image of wheat plant of size 1656*1356. The input image is divided into patches of size 256*256 before entering into the LPNet. The output of LPNet are patch-by-patch segmented mask images which are then combined to form the mask image as per the size of the input visual image. This image may contain some sort of inaccurate segmentation of the object (or, spikes) and are refined at global level using GMRNet network. The output of GMRNet network is nothing but the refined mask image containing spike regions only.

consists of 3 encoder blocks, and the output feature-maps of each encoder block are forwarded to the next encoder block for further feature extraction. Each encoder block consists of two convolution layers, each with the square filter of size 3*3 [26] with a varying number of filters (16, 64, 128) followed by ReLU [27] and max-pooling layer with a window size of 2*2 [28]. Square filters are popularly used in state-of-art methods [29], and the mentioned window size is considered as standard [13], [30]. Batch Normalization, a statistical procedure, is done to improve the performance as well as stability of the network. Input and output feature description of each encoder block in the Encoder_SpikeSegNet is presented in the tabular form (Table 1) and the algorithm for implementing the Encoder_SpikeSegNet network is given in Algorithm 1.

Decoder_SpikeSegNet network facilitates a special operation called transpose convolution [31], which up-sampled the incoming features to regenerate or decode the same. The resulting up-sampled feature maps are then concatenated/ merged with the corresponding encoded feature maps of the Encoder_SpikeSegNet. Merge operation helps in transferring the spatial information across the network for better localization of the segmented masks. The Decoder_SpikeSegNet contains three decoder blocks, and each decoder block consists of two convolution layers (with filter size 3*3) with a varying number of filters (128, 64, 16) as opposite to each encoder block in Encoder_SpikeSegNet and followed by ReLU operation to decode the features. The output of the final decoder was fed into the "SoftMax" [32] activation layer for classifying objects (or spikes). Input and output feature description of each decoder block in the Decoder_SpikeSegNet is presented in the tabular form (Table 2) and the algorithm for implementing the Decoder_SpikeSegNet network is given in Algorithm 2. Bottleneck_SpikeSegNet network contains three hourglasses, which provide more confident segmentation by concentrating the essential features captured at various occlusions, scale, and view-points [8], [13].

TABLE 1. Input and output feature description of each encoder block in the Encoder_SpikeSegNet Network.

Encoder Block #	Name of the Layers	Input feature size	# of kernel with size 3*3	Output feature size
Encoder Block-1	E_conv_1_1[p]	256*256*1	16	256*256*16
	E_conv_1_2[p]	256*256*16	16	256*256*16
	Pool-1	256*256*16	-	128*128*16
Encoder Block-2	E_conv_2_1[p]	128*128*16	64	128*128*64
	E_conv_2_2[p]	128*128*64	64	128*128*64
	Pool-2	128*128*64	-	64*64*64
Encoder Block-3	E_conv_3_1[p]	64*64*64	128	64*64*128
	E_conv_3_2[p]	64*64*128	128	64*64*128
	Pool-3	64*64*128	-	32*32*128

[p]Each convolution layer is followed by ReLU activation function and batch normalization
Feature size=x*y*z represents z number of features with x*y size
E_conv_u_v denotes the v^{th} convolution layer of the u^{th} encoder block number

TABLE 2. Input and output feature description of each decoder block in the Decoder_SpikeSegNet Network.

Decoder Block #	Name of the Layers	Input feature size	# of kernel with size 3*3	Output feature size
Decoder Block-1	T_conv-1[p]	32*32*128	128	64*64*128
	D_conv_1_1[q]	64*64*128	128	64*64*128
	D_conv_1_2[q]	64*64*128	128	64*64*128
Decoder Block-2	T_conv-2[p]	64*64*128	64	128*128*64
	D_conv_2_1[q]	128*128*64	64	128*128*64
	D_conv_2_2[q]	128*128*64	64	128*128*64
Decoder Block-3	T_conv-3[p]	128*128*64	16	256*256*16
	D_conv_3_1[q]	256*256*16	16	256*256*16
	D_conv_3_2[q]	256*256*16	16	256*256*16

[p]Transpose convolution operation followed by batch normalization and merge operation with the corresponding encoder block output
[q]Convolution operation followed by batch normalization

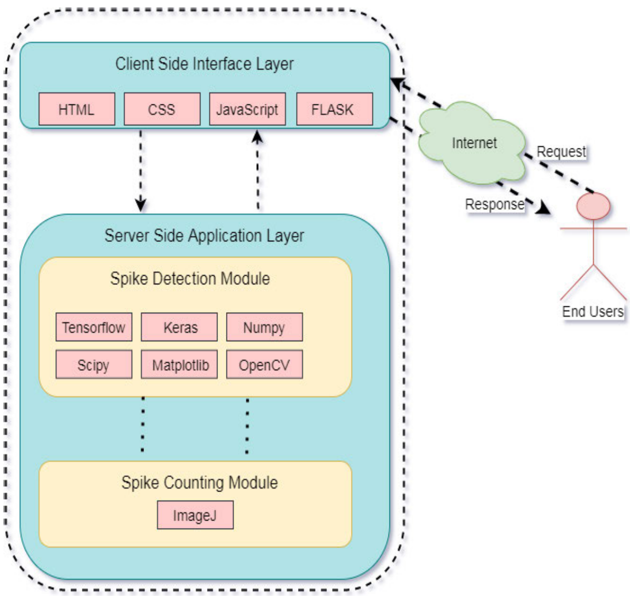


FIGURE 2. Architecture of Web-SpikeSegNet: The software architecture consists of two layers, namely Client-Side Interface Layer (CSIL) and Server Side Application Layer (SSAL). CSIL deals with the end-user's requests and its corresponding responses management. SSAL consists of two modules: spike detection and spike counting module.

Each hourglass comprises a sequence of residual blocks containing three convolution layers of filter size 1*1, 3*3, and 1*1 sequentially with depth (or the number of filters) 128, 128, and 256, respectively, estimated empirically on the basis of optimal performances. Algorithms for implementing Bottleneck_SpikeSegNet, LPNet, and GMRNet are presented in Algorithm 3, 4, and 5, respectively. The Spike counting module is integrated with the output of the Spike detection module in SSAL. For this purpose, the "Analyze Particle" functions of imageJ [33] was applied to the output image of GMRNet, which is a segmented mask image or binary image containing spike region only. "Analyze Particle" function implements a flood-fill technique [34] for counting of object.

B. TRAINING OF WEB-SpikeSegNet

For training the spike-detection module of Web-SpikeSegNet using the algorithms [1-5], 600 wheat plant's visual images were captured using the LemnaTec imaging facility installed at Nanaji Deshmukh Plant Phenomics Center, New Delhi, India. We have considered 3-directions (0^0, 120^0, 240^0) visual images w.r.t the initial position of the plant to overcome the problem of overlapping of ears. The image dataset was randomly divided into training and testing at 85% and 15%, respectively. Web-SpikeSegNet was trained for 300 epochs with batch size 32 due to the system platform constraints. Binary Cross-entropy loss function was used as it is a binary classification problem (i.e., pixels with either spike pixels or non-spike pixels) in the domain of image segmentation. Details of the hyper-parameters used to train the network are given in Table 3.

C. PERFORMANCE MEASUREMENT OF WEB-SpikeSegNet

For evaluating the segmentation performance to detect the spikes, the resulting segmented images (I^{pred}) using the Web-SpikeSegNet software are compared with the

Algorithm 1 Encode_SpikeSegNet: Encoding Operation of SpikeSegNet

I: Input image/feature
Conv(input feature, *filter*_size, no. of filters): Convolution operation ▷ for generating feature maps
BatchNorm(): Batch normalization operation ▷ for improving the performance as well as stability of the network
Pool(): Pooling operation or down-sampling with window size 2*2
procedure Encoder_SpikeSegNet(I) ▷ input image of size 256*256
    //First Encoder Block
    E_conv_1_1 ← *Conv*(I, 3 ∗ 3, 16) ▷ generates 16 feature maps of size 256*256
    E_batch_1_1 ← *BatchNorm*(E_conv_1_1) ▷ batch normalization of the features
    E_conv_1_2 ← *Conv*(E_batch_1_1, 3 ∗ 3, 16) ▷ generates 16 feature maps from the batch normalized features
    E_batch_1_2 ← *BatchNorm*(E_conv_1_2)
    I_Encoded_block_1 ← *Pool*(E_batch_1_2) ▷ size of each feature map reduced by half and returns 16 feature maps of size 128*128
    //Second Encoder Block. Here input is the output of First encoder block
    E_conv_2_1 ← *Conv*(I_Encoded_block_1, 3 ∗ 3, 64) ▷ generates 64 feature maps of size 128*128
    E_batch_2_1 ← *BatchNorm*(E_conv_2_1) ▷ batch normalization of the features
    E_conv_2_2 ← *Conv*(E_batch_2_1, 3 ∗ 3, 64)
    E_batch_2_2 ← *BatchNorm*(E_conv_2_2)
    I_Encoded_block_2 ← *Pool*(E_batch_2_2) ▷ return 64 feature maps of size 64*64
    //Third Encoder Block. Here input is the output of second encoder block
    E_conv_3_1 ← *Conv*(I_Encoded_block_2, 3 ∗ 3, 128) ▷ generates 128 feature maps of size 64*64
    E_batch_3_1 ← *BatchNorm*(E_conv_3_1)
    E_conv_3_2 ← *Conv*(E_batch_3_1, 3 ∗ 3, 128)
    E_batch_3_2 ← *BatchNorm*(E_conv_3_2)
    I_Encoded_block_3 ← *Pool*(E_batch_3_2) ▷ return 128 feature maps of size 32*32
return *I_Encoded_block_3*

TABLE 3. Hyper-parameters.

Optimizer	:	Adam
Learning rate	:	0.0005
Epoch	:	300
Batch size	:	32
Loss function	:	Binary Cross Entropy

corresponding ground-truth mask images (I^{grtr}), which were prepared by ensuing the steps mentioned in [3]. Segmentation performances are calculated using the following [Eq. (1) to Eq. (10)] statistical parameters [35]–[37]:

Type I Error (E_1): For any r^{th} test image, exclusive-OR operation is done to compute pixel-wise classification error ($Pix_E rr_r$) between (I^{pred}) and the corresponding (I^{grtr}) image of size p×q,

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

E_1 is computed by averaging the $Pix_E rr_r$ of all the test images:

$$E_1 = \frac{1}{n}\sum_{r=1}^{n} Pix_E rr_r \tag{2}$$

where, n is the total number of test 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^{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:

$$E_2^r = 0.5 * FPR + 0.5 * FNR \tag{3}$$

where,

$$FPR = \frac{1}{p*q}\sum_{l=1}^{q}\sum_{k=1}^{p}[(I^{\text{grtr}}(k,l).*I^{\text{pred}}(k,l)) \oplus I^{\text{pred}}(k,l)] \tag{4}$$

$$FNR = \frac{1}{p*q}\sum_{l=1}^{q}\sum_{k=1}^{p}[(I^{\text{grtr}}(k,l).*I^{\text{grtr}}(k,l)) \oplus I^{\text{pred}}(k,l)] \tag{5}$$

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

$$E_2 = \frac{1}{n}\sum_{r=1}^{n} E_2^r \tag{6}$$

Following performance parameters are also used for measuring the segmentation performance of the Web-SpikeSegNet at pixel level to identify/detect spikes as follows:

- True positive (TP): number of pixels correctly classified as spikes.
- True Negative (TN): number of pixels correctly classified as non-spikes (other than spike pixels).

Algorithm 2 Decoder_SpikeSegNet: Decoding Operation of SpikeSegNet

```
I: Output of Bottleneck_SpikeSegNet (for LPNet) or, output of Encoder_SpikeSegNet (for GMRNet).
Conv(input feature, filter_size, no. of filters): Convolution operation
BatchNorm(): Batch normalization operation
Tr_conv(input feature, filter_size, no. of filters): Transpose convolution   ▷ to up-sample the feature maps
Merge(): Merge/concatenation operation   ▷ for transferring the spatial information across the network
procedure Decoder_SpikeSegNet(I)   ▷ here input is 128 feature maps of size 32*32
    //First Decoder Block
    T_conv_1 ← Tr_Conv(I, 3 ∗ 3, 128)   ▷ Up-sampling done and return 128 decoded feature maps of size 64*64
    D_batch_1_1 ← BatchNorm(T_conv_1)   ▷ batch normalization of the features
    M_1 ← Merge(D_batch_1_1, I_Encoded_block_3)   ▷ concatenation operation with the output of third Encoder block [refer Algorithm 1 Line no.: 23]
    D_conv_1_1 ← Conv(M_1, 3 ∗ 3, 128)
    D_batch_1_2 ← BatchNorm(D_conv_1_1)   ▷ batch normalization of the features
    D_conv_1_2 ← Conv(D_batch_1_1, 3 ∗ 3, 128)
    I_Decoded_block_1 ← BatchNorm(D_conv_1_2)   ▷ Output of the 1st Decoder Block is 128 decoded feature maps of size 64*64
    //Second Decoder Block. Here input is the output of First Decoder block
    T_conv_2 ← Tr_Conv(I_Decoded_block_1, 3 ∗ 3, 64)   ▷ Up-sampling done and return 64 decoded feature maps of size 128*128
    D_batch_2_1 ← BatchNorm(T_conv_2)   ▷ batch normalization of the features
    M_2 ← Merge(D_batch_2_1, I_Encoded_block_2)   ▷ concatenation operation with the output of second Encoder block [refer Algorithm 1 Line no.: 17]
    D_conv_2_1 ← Conv(M_2, 3 ∗ 3, 64)
    D_batch_2_2 ← BatchNorm(D_conv_2_1)
    D_conv_2_2 ← Conv(D_batch_2_2, 3 ∗ 3, 64)
    I_Decoded_block_2 ← BatchNorm(D_conv_2_2)   ▷ Output of the second Decoder Block is 64 decoded feature maps of size 128*128
    //Third Decoder Block. Here input is the output of Second Decoder block
    T_conv_3 ← Tr_Conv(I_Decoded_block_2, 3 ∗ 3, 16)   ▷ Up-sampling done and return 16 decoded feature maps of size 256*256
    D_batch_3_1 ← BatchNorm(T_conv_3)   ▷ batch normalization of the features
    M_3 ← Merge(D_batch_3_1, I_Encoded_block_1)   ▷ concatenation operation with the output of First Encoder block [refer Algorithm 1 Line no.: 11]
    D_conv_3_1 ← Conv(M_3, 3 ∗ 3, 16)
    D_batch_3_2 ← BatchNorm(D_conv_3_1)
    D_conv_3_2 ← Conv(D_batch_3_2, 3 ∗ 3, 16)
    I_Decoded_block_3 ← BatchNorm(D_conv_3_2)▷ Output of the third Decoder Block is 16 decoded feature maps of size 256*256
return I_Decoded_block_3
```

- False Positive (FP): number of non-spike pixels classified as spikes pixels.
- False Negative (FN): number of spike pixels classified as non- spikes pixels.

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

$$Precision = TP/(TP + FP) \tag{7}$$

measures the percentage of detected pixels are actually spikes

$$Recall = TP/(TP + FN) \tag{8}$$

measures the percentage of actually spikes spike pixels are detected

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{9}$$

measures performance of the Web-SpikeSegNet

$$F_1Score = 2(Precision * Recall)/(Precision + Recall) \tag{10}$$

measures robustness of the Web-SpikeSegNet in detecting or identifying spikes

III. RESULTS AND DISCUSSION

To demonstrate the working environment of Web-SpikeSegNet, a case study is presented here. The architecture of Web-SpikeSegNet mentioned in section 3, and the design of the software consists of 5 sections, namely "Home page", "Spike Detection and Counting", "Help", "Contact Us", and "Sample Data set". The "Home page" contains basic

Algorithm 3 Bottleneck_SpikeSegNet

```
I: Input image/feature
Conv(input feature, filter_size, no. of filters): Convolution operation
BatchNorm(): Batch normalization operation
Tr_conv(input feature, filter_size, no. of filters): Transpose convolution operation
Pool(): Pooling operation or down-sampling with window size 2*2
Merge(): Merge/concatenation operation
procedure Bottleneck_SpikeSegNet(I) ▷ here, input is output of ENCODER_SPIKESEGNET, 128 feature maps of size 32*32
    H_1 ← HOURGLASS_SPIKESEGNET (I)    ▷ Call HOURGLASS_SPIKESEGNET procedure and return, 128 feature maps of size 32*32
    Scale_up_ ← SCALE_UP(H_1)    ▷ Call SCALE_UP procedure and return, 128 feature maps of size 64*64
    H_2 ← HOURGLASS_SPIKESEGNET (Scale_up)
    Scale_down_ ← SCALE_DOWN (H_2)    ▷ Call SCALE_DOWN procedure and return, 128 feature maps of size 32*32
    H_3_ ← HOURGLASS_SPIKESEGNET(Scale_down)
return H_3    ▷ return, 128 refined feature maps of size 32*32
    ▷ Hourglass gives more confident segmentation by concentrating on the essential features
procedure Hourglass_SpikeSegNet(I)
    res_1 ← RESIDUAL_BL(I)    ▷ returns, 256 feature maps of size 32*32
    pool_1 ←Pool(res_1)    ▷ down-sampling done and returns, 256 feature maps of size 16*16
    res_2 ← RESIDUAL_BL(pool_1)    ▷ returns, 256 feature maps of size 16*16
    pool_2 ←Pool(res_2)    ▷ down-sampling done and returns, 256 feature maps of size 8*8
    res_3 ← RESIDUAL_BL(pool_2)    ▷ returns, 256 feature maps of size 8*8
    pool_3 ←Pool(res_3)    ▷ down-sampling done and returns, 256 feature maps of size 4*4
    res_4 ← RESIDUAL_BL(pool_3)    ▷ returns, 256 feature maps of size 4*4
    res_5 ← RESIDUAL_BL(res_4)
    T_conv_1 ←Tr_conv(res_5, 3 ∗ 3, 256)    ▷ up-sampling done and returns, 256 feature maps of size 8*8
    M_1 ←Merge(T_conv_1, res_3)
    res_6 ← RESIDUAL_BL(M_1)    ▷ returns, 256 feature maps of size 8*8
    T_conv_2 ←Tr_conv(res_6, 3 ∗ 3, 256)    ▷ up-sampling done and returns, 256 feature maps of size 16*16
    M_2 ←Merge(T_conv_2, res_2)
    res_7 ← RESIDUAL_BL(M_2)    ▷ returns, 256 feature maps of size 16*16
    T_conv_3 ←Tr_conv(res_7, 3 ∗ 3, 256)    ▷ up-sampling done and returns, 256 feature maps of size 32*32
    M_3 ←Merge(T_conv_3, res_1)
    res_8 ← RESIDUAL_BL(M_3)    ▷ returns, 256 feature maps of size 32*32
return res_8
procedure Residual_bl(I)
    res_conv_1 ←Conv(I, 1 ∗ 1, 128)
    res_conv_2 ←Conv(res_conv_1, 3 ∗ 3, 128)
    res_conv_3 ←Conv(res_conv_2, 1 ∗ 1, 256)
return res_conv_3    ▷ returns, 256 feature maps    ▷ Scale up and scale down operations help in finding the relationships among aggregate features at different scales which further helps in getting the robust features
procedure Scale_up(I)
    sc_up_conv_1 ←Conv(I, 3 ∗ 3, 128)
    sc_up_batch_1 ←BatchNorm(sc_up_conv_1)
    sc_up_conv_2 ←Conv(sc_up_batch_1, 3 ∗ 3, 128)
    sc_up_batch_2 ←BatchNorm(sc_up_conv_2)
    sc_up_pool ←Tr_Pool(sc_up_batch_2)
return sc_up_pool
procedure Scale_down(I)
    sc_down_pool_1 ←Pool(I)
    sc_down_conv_1 ←BatchNorm(sc_down_pool_1, 3 ∗ 3, 128)
    sc_down_batch_1 ←BatchNorm(sc_down_conv_1)
    sc_down_conv_2 ←Conv(sc_down_batch_1, 3 ∗ 3, 128)
    sc_down_batch_2 ←BatchNorm(sc_down_conv_2)
return sc_down_batch_2
```

Algorithm 4 LPNet Local Patch Extraction Network

1: **I**: Input image/feature
2: **procedure** LPNet(I) ▷ here input is visual image patches of size 256*256
3: $Encoded\_I \leftarrow$ ENCODER_SPIKESEGNET(I) ▷ Call Algorithm 1. Return encoded feature maps of the input image
4: $Bottleneck\_I \leftarrow$ BOTTLENECK_SPIKESEGNET($Encoded\_I$) ▷ Call Algorithm 3. Return refined feature maps of the input features
5: $Decoded\_I \leftarrow$ DECODER_SPIKESEGNET($Bottleneck\_I$) ▷ Call Algorithm 2. Return decoded feature maps of the input features
6: **return** $Decoded\_I$ ▷ Segmeted mask image of size 256*256 containing spikes regions corresponding to the input patches.

Algorithm 5 GMRNet

1: **I**: Input image/feature
2: **procedure** GMRNet(I) ▷ here input is the output image/feature of LPNet
3: $Encoded\_I \leftarrow$ ENCODER_SPIKESEGNET(I) ▷ Call Algorithm 1. Return encoded feature maps of the input image
4: $Decoded\_I \leftarrow$ DECODER_SPIKESEGNET($Encoded\_I$) ▷ Call Algorithm 2. Return decoded feature maps of the input features
5: **return** $Decoded\_I$ ▷ Refined segmented mask image of size 256*256 containing spikes regions corresponding to the input image/feature.

TABLE 4. Segmentation performance analysis of Web-SpikeSegNet.

Type I Error	Type II Error	Accuracy	Precision	Recall	F1 Score
0.00159	0.0586	0.9965	0.9959	0.9961	0.9965

information about SpikeSegNet, and the flow diagram of the steps needs to be followed to recognize and count the spikes of the uploaded wheat plant image (Fig. 3). The ''Sample Data set'' section facilitates sample visual images of wheat plants for the experiment. Spike Detection and Counting module is the center of attention of the software. The user has to follow the following steps to detect and count the spikes and the output of each steps are pictorially presented in Supplementary 1:

1) Select and upload visual image of wheat plant of size 1656*1356 consisting of above ground parts only as discussed in [3].
2) Click on ''Generate Patches'' button for dividing the whole image into patches. Here, the visual image is divided into 100 pixel overlapping patches (each patches of size 256*256) which work as input to the LPNet module. Therefore, from one visual image of size 1656*1356, 180 patches of size 256*256 will be generated.
3) Click on ''Run LPNet'' to run the LPNet module for extracting contextual and spatial features at patch level. Output of the LPNet are the segmented images of size 256*256 corresponding to the patch images.
4) The output of LPNet are merged to generate the segmented image of size 1656*1656 that contains some inaccurate segmentation of spikes and further refined at global level by clicking on ''Run GMRNet'' button.
5) For counting the wheat spikes, click on ''Count''button and the corresponding spikes count will be displayed on the next window.

The final output of Web-SpikeSegNet after detection and counting of spikes from the visual images of wheat plant is given in Fig. 4.

A. PERFORMANCE ANALYSIS OF WEB-SpikeSegNet

Web-SPikeSegNet was trained using the training dataset consisting of randomly selected 85% of the total images captured (i.e., 510 images among 600 images). Although the network was trained for 300 epochs, the training losses were plateaued around 100 epoch as given in Fig 5. Segmentation performances of the Web-SpikeSegNet has been computed on the testing dataset consists of 90 images. The mentioned statistical parameters (eq. 1 to eq. 10) are computed, and the average values are presented in Table 4. As the performance of spike detection is calculated at the pixel level, the value of E1 (=0.00159) depict that on an average only 104 pixels are misclassified among 65,536 pixels which is the pixel size of one image, i.e., 65,536 (256 * 256). The accuracy of the approach as well as the developed software is around 99.65 %. The average precision value reflects that 99.59% of the detected spikes are actually spike pixels and the robustness of the approach is also $\sim$ 100 %.

B. COMPARATIVE ANALYSIS WITH ACID (Annotated CROP IMAGE Dataset) DATASET AVAILABLE AT https://plantimages.nottingham.ac.uk/

We ran the developed software on the ACID (Annotated Crop Image Dataset) dataset for the comparative study. The dataset consisted of 415 training images and 105 testing images and was contributed by Pound *et al.* [6]. They proposed a multi-task deep learning architecture for localizing

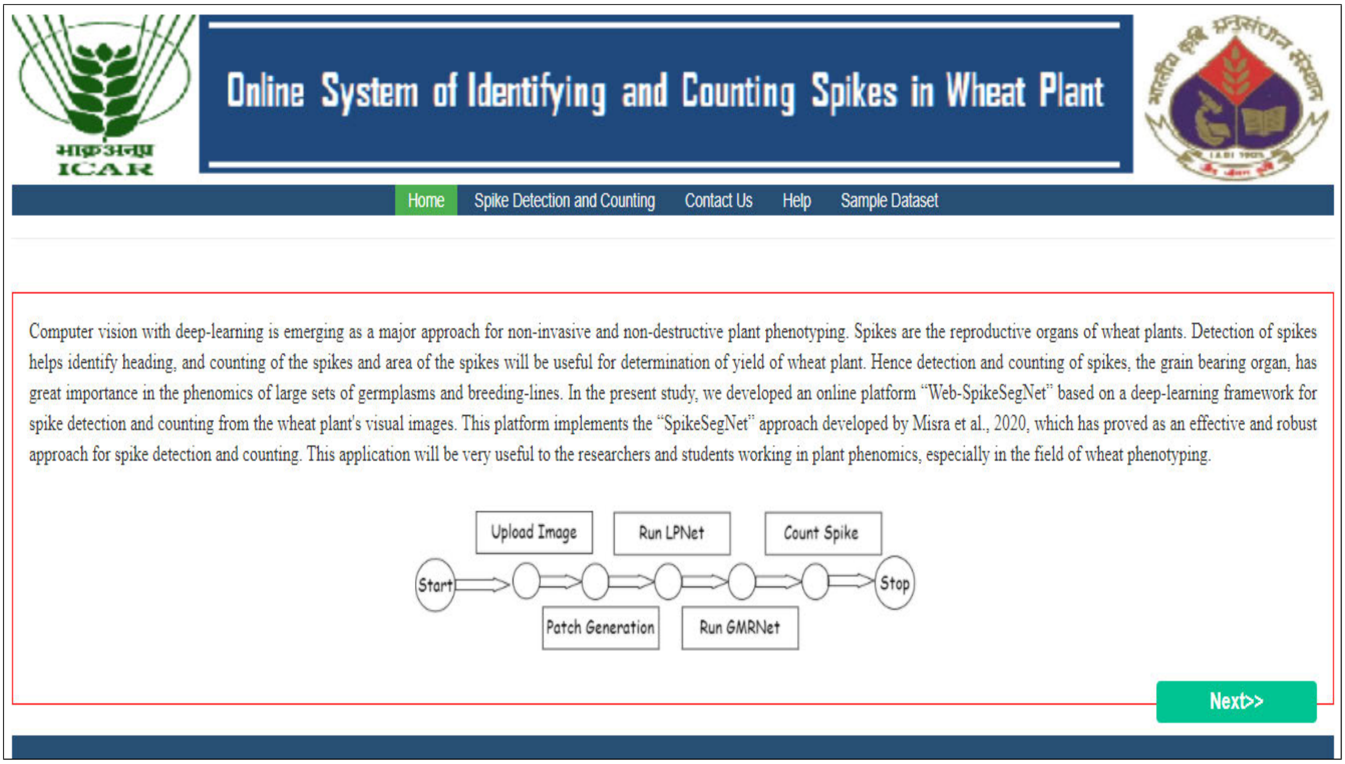


FIGURE 3. Home page of Web-SpikeSegNet contains basic information about SpikeSegNet and the flow diagram of the steps need to be followed to recognize and counting the spikes of the uploaded wheat plant image.

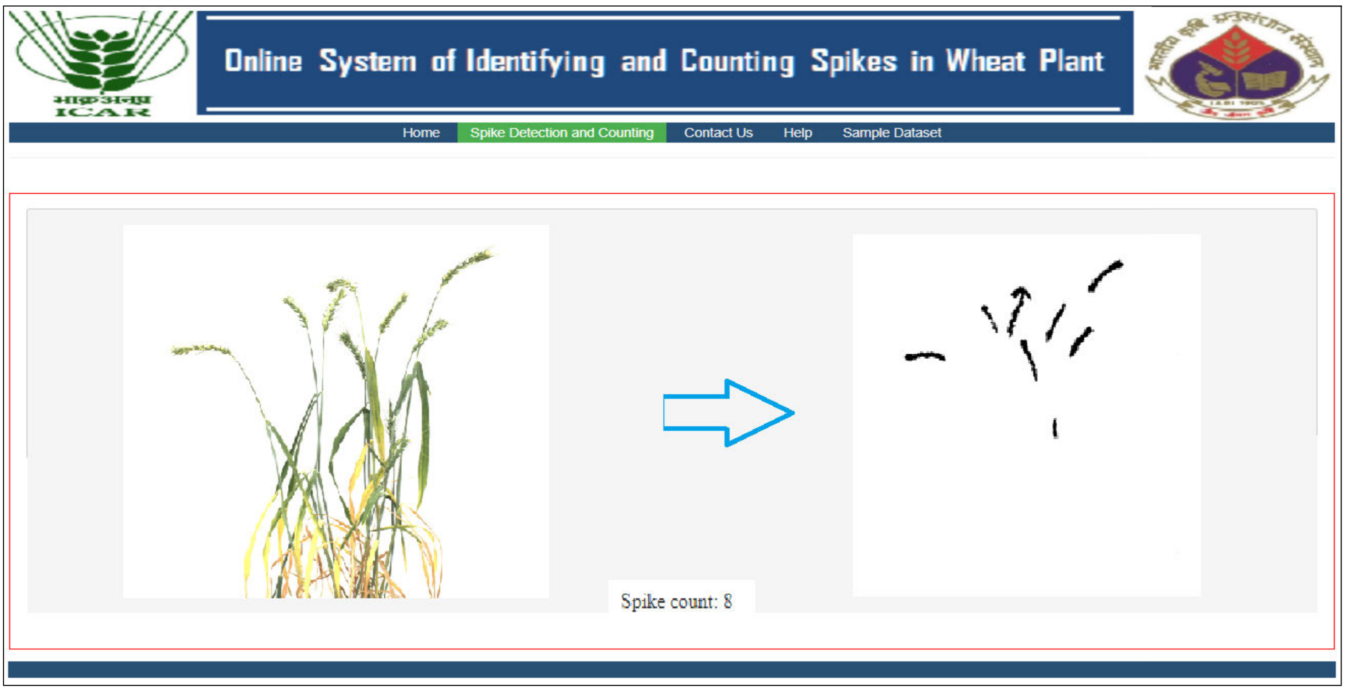


FIGURE 4. The final output of Web-SpikeSegNet after detection and counting of spikes from the visual images of wheat plant.

wheat spikes and spikelets and achieved 95 % accuracy in spike detection. As the Web-SpikeSegNet model was trained using the wheat's visual images with a consistent white background, we converted the background of the test images in the mentioned website from black to white to conduct the comparative study. The output of Web-SpikeSegNet on

TABLE 5. Segmentation performance analysis of the ACID dataset.

Type I Error	Type II Error	Accuracy	Precision	Recall	F1 Score
0.00164	0.0576	0.9955	0.9962	0.9958	0.9962

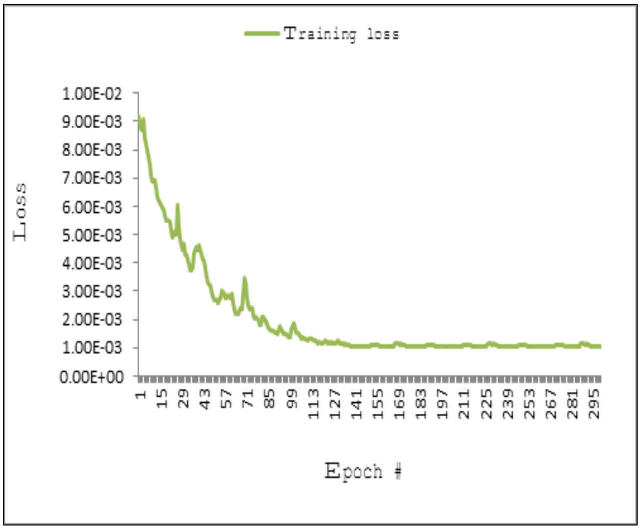


FIGURE 5. Graphical representation of training Loss.

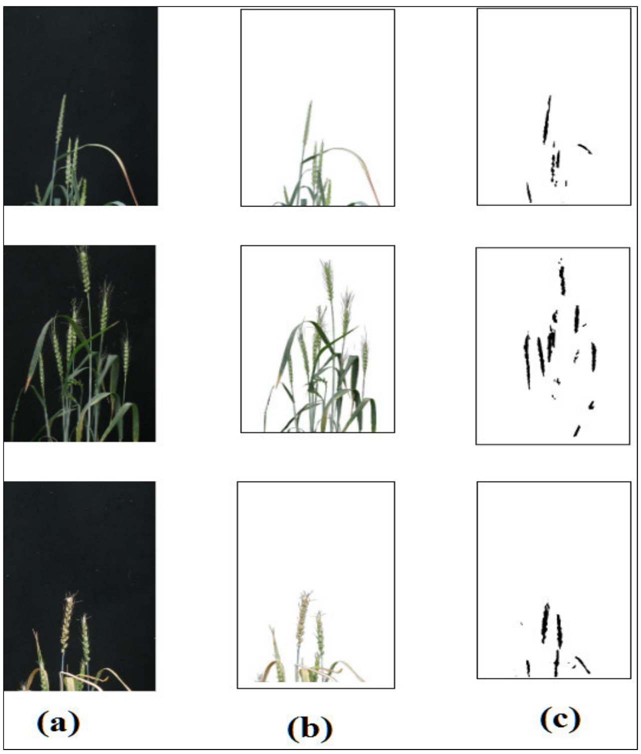


FIGURE 6. Comparative study with ACID (Annotated Crop Image Dataset) dataset available at https://plantimages.nottingham.ac.uk/: (a) test images (b) black background converted into white (c) detected spikes using Web-SpikeSegNet software.

the ACID dataset is presented in Fig. 6. For computing the segmentation performance, the ground-truth mask images corresponding to the testing dataset were prepared using the procedure mentioned in [3]. The average segmentation performances are given in Table 5. The value of the type I error (0.00164) reflects that, on average, only 107 pixels are wrongly classified among 65,536 pixels (the size of one image is 256*256 pixels). The accuracy (99.55%), precision (99.62%), and F1 value (99.62%) depict that the Web-SpikeSegNet approach is comparatively generalized and robust than the approach presented by Pound *et al.* [6]. It is due to the training criteria of Web-SpikeSegNet, where the deep learning model is trained at patch level for understanding the local as well as global features efficiently.

IV. CONCLUSION

Recognition and counting of spikes for the large set of germplasms in a non-destructive way is an enormously challenging task. This study developed web-based software "Web-SpikeSegNet" using the robust SpikeSegNet approach, which is based on digital image analysis and deep-learning techniques. The software is freely available for researchers, and students are working particularly in the field of wheat plant phenotyping. Further, it is a useful tool in the automated phenomics facility to automate the phenology-based treatment. Web-SpikeSegNet is a significant step toward studying the wheat crop yield phenotyping and can be extended to the other cereal crops.

ACKNOWLEDGMENT

Tanuj Misra acknowledges the Nanaji Deshmukh Plant Phenomics Facility, ICAR-IARI, New Delhi, for the facilities. Dr. Jahangeer A. Bhat from RLBCAU-Jhansi is highly acknowledged for assisting in the rebuttal preparation.

REFERENCES

[1] J. R. Porter and S. Christensen, "Deconstructing crop processes and models via identities," *Plant, cell Environ.*, vol. 36, no. 11, pp. 1919–1925, Nov. 2013.
[2] S. A. Tsaftaris, M. Minervini, and H. Scharr, "Machine learning for plant phenotyping needs image processing," *Trends Plant Sci.*, vol. 21, no. 12, pp. 989–991, Dec. 2016.
[3] T. Misra, A. Arora, S. Marwaha, V. Chinnusamy, A. R. Rao, R. Jain, R. N. Sahoo, M. Ray, S. Kumar, D. Raju, R. R. Jha, A. Nigam, and S. Goel, "SpikeSegNet—A deep learning approach utilizing encoder-decoder network with hourglass for spike segmentation and counting in wheat plant from visual imaging," *Plant Methods*, vol. 16, no. 1, pp. 1–20, Dec. 2020.
[4] K. Bi, P. Jiang, L. Li, B. Shi, and C. Wang, "Non-destructive measurement of wheat spike characteristics based on morphological image processing," *Trans. Chin. Soc. Agricult. Eng.*, vol. 26, no. 12, pp. 212–216, 2010.
[5] L. Qiongyan, J. Cai, B. Berger, M. Okamoto, and S. J. Miklavcic, "Detecting spikes of wheat plants using neural networks with laws texture energy," *Plant Methods*, vol. 13, no. 1, p. 83, Dec. 2017.
[6] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French, "Deep learning for multi-task plant phenotyping," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 2055–2063.
[7] P. Sadeghi-Tehran, K. Sabermanesh, N. Virlet, and M. J. Hawkesford, "Automated method to determine two critical growth stages of wheat: Heading and flowering," *Frontiers Plant Sci.*, vol. 8, p. 252, Feb. 2017.

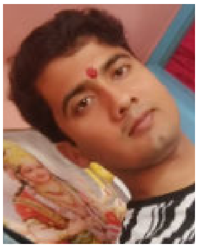
[8] M. M. Hasan, J. P. Chopin, H. Laga, and S. J. Miklavcic, ''Detection and analysis of wheat spikes using convolutional neural networks,'' *Plant Methods*, vol. 14, no. 1, p. 100, Dec. 2018.
[9] E. David, S. Madec, P. Sadeghi-Tehran, H. Aasen, B. Zheng, S. Liu, N. Kirchgessner, G. Ishikawa, K. Nagasawa, M. A. Badhon, C. Pozniak, B. de Solan, A. Hund, S. C. Chapman, F. Baret, I. Stavness, and W. Guo, ''Global wheat head detection (GWHD) dataset: A large and diverse dataset of high-resolution RGB-labelled images to develop and benchmark wheat head detection methods,'' *Plant Phenomics*, vol. 2020, pp. 1–12, Aug. 2020.
[10] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.
[11] H. Xiong, Z. Cao, H. Lu, S. Madec, L. Liu, and C. Shen, ''TasselNetv2: In-field counting of wheat spikes with context-augmented local regression networks,'' *Plant Methods*, vol. 15, no. 1, pp. 1–14, Dec. 2019.
[12] P. Sadeghi-Tehran, N. Virlet, E. M. Ampe, P. Reyns, and M. J. Hawkesford, ''DeepCount: In-field automatic quantification of wheat spikes using simple linear iterative clustering and deep convolutional neural networks,'' *Frontiers Plant Sci.*, vol. 10, p. 1176, Sep. 2019.
[13] O. Ronneberger, P. Fischer, and T. Brox, ''U-Net: Convolutional networks for biomedical image segmentation,'' in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, Oct. 2015, pp. 234–241.
[14] V. Badrinarayanan, A. Kendall, and R. Cipolla, ''SegNet: A deep convolutional encoder-decoder architecture for image segmentation,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[15] R. R. Jha, G. Jaswal, D. Gupta, S. Saini, and A. Nigam, ''PixISegNet: Pixel-level iris segmentation network using convolutional encoder–decoder with stacked hourglass bottleneck,'' *IET Biometrics*, vol. 9, no. 1, pp. 11–24, Jan. 2020.
[16] T. Berners-Lee, ''Tim berners-lee,'' Bloomberg Businessweek, 1989.
[17] E. A. Meyer, *Cascading Style Sheets: The Definitive Guide*. Newton, MA, USA: O'Reilly Media, 2004.
[18] G. Mainland, M. Welsh, and G. Morrisett, ''Flask: A language for data-driven sensor network programs,'' Harvard Univ., Cambridge, MA, USA, Tech. Rep. TR-13-06, 2006.
[19] S. Yehuda and S. Tomer, ''Advanced javascript programming,'' BPB, New Delhi, India, 1998.
[20] T. Hope, Y. S. Resheff, and I. Lieder, *Learning TensorFlow: Guide to Building Deep Learning Systems*. Newton, MA, USA: O'Reilly Media, 2017.
[21] A. Gulli and S. Pal, *Deep Learning With Keras*. Birmingham, U.K.: Packt, 2017.
[22] E. Bressert, *SciPy and NumPy: An Overview for Developers*. Newton, MA, USA: O'Reilly Media, 2012.
[23] E. A. Christensen, F. J. Blanco-Silva, *Learning SciPy for Numerical and Scientific Computing*. Birmingham, U.K: Packt, 2015.
[24] S. Tosi, *Matplotlib for Python Developers*. Birmingham, U.K.: Packt, 2009.
[25] J. Howse, *OpenCV Computer Vision With Python*. Birmingham, U.K.: Packt, 2013.
[26] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, ''A convolutional neural network for modelling sentences,'' 2014, *arXiv:1404.2188*. [Online]. Available: http://arxiv.org/abs/1404.2188
[27] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, ''Learning activation functions to improve deep neural networks,'' 2014, *arXiv:1412.6830*. [Online]. Available: http://arxiv.org/abs/1412.6830
[28] B. Graham, ''Fractional max-pooling,'' 2014, *arXiv:1412.6071*. [Online]. Available: http://arxiv.org/abs/1412.6071
[29] K. Simonyan and A. Zisserman, ''Very deep convolutional networks for large-scale image recognition,'' 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[30] S. P. Mohanty, D. P. Hughes, and M. Salathé, ''Using deep learning for image-based plant disease detection,'' *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016.
[31] V. Dumoulin and F. Visin, ''A guide to convolution arithmetic for deep learning,'' 2016, *arXiv:1603.07285*. [Online]. Available: http://arxiv.org/abs/1603.07285
[32] W. Liu, Y. Wen, Z. Yu, and M. Yang, ''Large-margin softmax loss for convolutional neural networks,'' in *Proc. ICML*, 2016, vol. 2, no. 3, p. 7.
[33] M. D. Abràmoff, P. J. Magalhães, and S. J. Ram, ''Image processing with ImageJ,'' *Biophoton. Int.*, vol. 11, no. 7, pp. 36–42, 2004.
[34] A. Asundi and Z. Wensen, ''Fast phase-unwrapping algorithm based on a gray-scale mask and flood fill,'' *Appl. Opt.*, vol. 37, no. 23, pp. 5416–5420, 1998.
[35] H. Proenca, S. Filipe, R. Santos, J. Oliveira, and L. A. Alexandre, ''The UBIRIS.v2: A database of visible wavelength iris images captured on-the-move and at-a-distance,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1529–1535, Aug. 2010.
[36] M. Haindl and M. Krupička, ''Unsupervised detection of non-iris occlusions,'' *Pattern Recognit.*, vol. 57, pp. 60–65, May 2015.
[37] Z. Zhao and A. Kumar, ''An accurate iris segmentation framework under relaxed imaging constraints using total variation model,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3828–3836.

TANUJ MISRA received the Ph.D. degree in computer application from ICAR-IARI, New Delhi. He is currently working as a Teacher cum Research Associate in computer science with Rani Lakshmi Bai Central Agricultural University, Jhansi. He has authored many research articles in national/international journals of repute. His research interests include applications of machine learning, deep learning, data mining, image analysis, especially on RGB, IR, and NIR sensors, and techniques in plant phenomics domain.

ALKA ARORA received the Ph.D. degree in data mining with a focus on cluster characterization using rough set theory.

She is currently working as a Principal Scientist with the Indian Agricultural Statistics Research Institute (IASRI), which is a premier institute in the area of statistics, computer applications, and bio-informatics under the Indian Council of Agricultural Research (ICAR). She is also a Research Guide and a Faculty Member of the PG School, IARI, New Delhi, which has the status of an International University (Deemed). She has more than 23 years of experience in research, teaching, and training. She contributed toward strengthening of agricultural informatics through designing and implementing number of e-governance applications, Web applications, and mobile applications operational at National level. Her students are pursuing research in the areas of deep learning and data mining using image processing data. Her research interests include applications of machine learning, deep learning, data mining, image analysis techniques for agriculture domain along with development of decision support system, and mobile applications for e-governance.

SUDEEP MARWAHA received the B.Sc. degree in electronics from the University of Delhi, New Delhi, India, in 1995, the M.Sc. degree in computer applications from the Indian Agricultural Research Institute, New Delhi, and the Ph.D. degree in computer science from the University of Delhi, in 2008. He is currently a Principal Scientist and a Professor with the Division of Computer Applications, ICAR-IASRI, New Delhi. He completed following projects solution architect for semantic Web enabled systems, management information systems, ERP (Oracle Apps), knowledge base systems, and image analysis based systems. He has published more than 50 research articles. His research interests include artificial intelligence, semantic Web, and ontologies.



RANJEET RANJAN JHA received the master's degree in computer science from the Department of Computer Science and Engineering, NIT Goa. He is currently pursuing the Ph.D. degree with the School of Computing and Electrical Engineering, IIT Mandi. He has authored many research articles in national/international journals of repute. His research interests include medical image analysis, biometrics, deep learning, computer vision, and security.

MRINMOY RAY received the Ph.D. degree in agricultural statistics from the ICAR-Indian Agricultural Research Institute, New Delhi, India, in 2017. He is currently working as a Scientist with the Division of Forecasting and Agricultural Systems Modeling, ICAR-Indian Agricultural Statistics Research Institute, New Delhi. He has got more than 35 publications to his credit in nationally and internationally reputed journals. His current research interests include time series modeling, technology forecasting, and machine learning.

RAJNI JAIN received the Ph.D. degree in computer science from Jawaharlal Nehru University, New Delhi, India, in 2005. She is currently a Principal Scientist with ICAR-NIAP and a Faculty Member with the Post Graduate School, ICAR-IARI for teaching and guiding students of M.Sc. and Ph.D. in computer application. She has organized four 21 days training programs and many short duration training programs for the capacity development of national agricultural research systems. She has published more than 100 research articles in reputed national and international journals. Her research interests include data mining, development of optimal crop plans using evolutionary computing, deep learning, image processing, plant disease diagnosis, rough sets, decision tree induction, classification techniques, machine learning, and agricultural policy research.

A. R. RAO has been a Principal Scientist and a Professor of bioinformatics with the ICAR-Indian Agricultural Statistics Research Institute, and a fellow of the National Academy of Agricultural Sciences (NAAS), New Delhi, since January 2016. He is also a fellow of the Indian Society of Genetics and Plant Breeding, New Delhi, and the Society for Applied Biotechnology, India. He completed more than 30 international and national funded projects. He has published more than 150 research articles. His research interests include bioinformatics and computational biology, statistical genetics, and genomics.

ELDHO VARGHESE received the Ph.D. degree (Hons.) in agricultural statistics from the Indian Agricultural Research Institute, New Delhi. He is currently working as a Scientist (Senior Scale) with the Central Marine Fisheries Research Institute, Kochi, Kerala, (formerly, Indian Agricultural Statistics Research Institute, New Delhi). He has developed several packages and Web solutions for generation of statistical designs. He has published around 150 research articles, 83 abstracts in various conferences, one book, one policy series document, 23 reference manual/technical bulletins, five book chapters, nine e-manuals, and several popular articles. His main research interests include statistical designs for competition effects, mating designs for plant breeding trials, statistical designs for product and process optimization, fish stock assessment, statistical modeling in fisheries, and remote sensing applications in fisheries. He was awarded with the prestigious Lal Bahadur Shastri Outstanding Young Scientist Award—2017 from ICAR, India; the NAAS Young Scientist Award from the National Academy of Agricultural Sciences, India, in the field of social sciences for the biennium 2017 to 2018; and the IARI Gold Medal for his Ph.D. degree.

SHAILENDRA KUMAR received the Ph.D. degree in statistics from the University of Lucknow, Lucknow, India, in 2018. He is currently working as a Teaching cum Research Associate with the Department of Computer Science, College of Horticulture and Forestry, Rani Lakshmi Bai Central Agricultural University, Jhansi. His research interests include Bayesian inference, machine learning, and survival analysis. He received the Chancellor Silver Medal for his M.Phil. degree.

SUDHIR KUMAR received the Ph.D. degree in plant physiology from ICAR-IARI, New Delhi. He served with the Indian Institute for Farming System Research, Modipuram, and the Indian Institute of Agricultural Research, New Delhi. He is currently working as a Scientist in plant physiology with the Indian Agricultural Research Institute. He is also associated with teaching for PG students for plant physiology courses. He is also working on RGB, IR, NIR, and Crop reporter sensors with the Nanaji Deshmukh Plant Phenomics Centre for drought and nutrient tolerance in wheat in high throughput manner with the help of the imaging sensors under artificial environmental conditions for developing climate resilient varieties. He has authored many research articles in national/international journals of repute. He has published training manuals, technical bulletins popular articles, and books. He has developed new course on phenomics for PG student. He has organized several trainings and workshop on image-based phenotyping for faculty and students. His main research interests include abiotic stress, plant phenotyping, cropping system management, image based phenotyping, and sensor-based plant trait dissection.

ADITYA NIGAM received the M.Tech. and Ph.D. degrees from IIT Kanpur, in 2009 and 2014, respectively. In August 2014, he joined SCEE, as a Teaching Fellow. He is currently working as an Assistant Professor with the School of Computing and Electrical Engineering (SCEE), IIT Mandi. He has authored many research articles in national/international journals of repute. He has designed deep-learning courses and published related training manuals. He is also working in the areas of biometrics, image processing, computer vision, and machine learning.

RABI NARAYAN SAHOO is currently a Principal Scientist with the Division of Agricultural Physics, CAR-Indian Agricultural Research Institute, Library Avenue, New Delhi, India. He associated with many multidisciplinary and multi-institutional collaborative national and international research projects on remote sensing research programs, as PI or Co-PI with total funding of INR 105 crores over last 19 years. His major accomplishments in applications of remote sensing and GIS in agriculture includes monitoring biotic and abiotic stresses for assessing crop conditions, remote sensing-based drought monitoring and early warning, quantitative assessment of soil attributes for fertility and quality assessment, modeling for site specific nutrient requirement, precision agriculture, image and spectral-based high throughput plant phenotyping, targeting resource conserving technologies, technologies to reduce agricultural fallow lands through biophysical monitoring, and identifying production constraints of marginal lands and land use cover change modeling. He has published more than 180 research articles. His major research interests include hyperspectral remote sensing for soil and crop health monitoring for precision farming, plant phenomics, and UAV remote sensing.

VISWANATHAN CHINNUSAMY received the B.Sc. degree from the Agricultural College and Research Institute, TNAU, Killikulam, in 1990, and the M.Sc. and Ph.D. degrees from the ICAR-Indian Agricultural Research Institute, New Delhi, in 1993 and 1996, respectively. He was a Senior Scientist, from 1996 to 2010, and a BOYSCAST Fellow, from 2001 to 2002. From 2008 to 2010, he was a Visiting Researcher with the Department of Botany and Plant Sciences, University of California. In 2009, he was a Visiting Scholar with the Department of Molecular Biology, Harvard Medical School, Harvard University, Boston. He has been a Principal Scientist, since 2010, and the Head of the Division of Plant Physiology, ICAR-Indian Agricultural Research Institute, New Delhi, since 2013. He is a fellow of the Indian Society for Plant Physiology. His awards/honours include the ICAR Jawaharlal Nehru Award, in 2001; the DST BOYSCAST Fellowship Award, in 2001; and the ISPP J.J. Chinoy Gold Medal.