A Deep-Learning Phenotypic Model to Estimate Key Yield related Traits in Field Conditions for UK Bread Wheat

Tahani Alkhudaydi

A thesis presented for the degree of

Doctor of Philosophy

School of Computer Science

University of East Anglia

United Kingdom

September 2020

A Deep-Learning Phenotypic Model to Estimate Key Yield related Traits in Field Conditions for UK Bread Wheat

Tahani Alkhudaydi

September 2020

Abstract

Wheat is one of the three major crops in the world with a total demand expected to exceed 850 million tons by 2050. One of the key challenges for wheat is to stabilise the yield and quality in wheat production. The application of the internet of things (IoT) in agriculture has enabled us to continuously monitor crop growth through networked remote sensors and non-invasive imaging devices. Analysis of the output of such systems with machine-learning algorithms and image processing techniques can help to extract meaningful information to assisting crop management.

Counting wheat spikelets from infield images is considered one of the challenges related to estimating yield traits of wheat crops. For this challenging problem, we first propose a model for semantic segmentation, SpikeSEG, to isolate the spike regions from noisy and redundant background. Segmentation results can then be used to improve phenotypic counting approaches. In-field spikelet counting is very challenging because of spikelet self-similarity, high volume in one image, and severe occlusion. For this, we propose a density estimation approach related to crowd counting, SpikeCount. Our proposed segmentation/counting methods are based on deep learning architectures as those have the advantage of being able to identify features automatically.

Annotation of images with the ground truth are required for machine learning approaches, but those are expensive in terms of time and resources. We use Transfer Learning in both tasks, segmentation and counting. We also investigated the ef-

fect of multi-task learning by using SpikeMulti and compared it with conventional workflow of applying segmentation and the density estimation.

Our results indicate the segmentation is beneficial as focusing only on the regions of interest improves counting accuracy in most scenarios. In particular, a combination of Transfer Learning and Multi-task learning produced excellent results for the counting task for most of the stages of wheat development.

Supervisors: Beatriz De La Iglesia and Michal Mackiewicz

A Deep-Learning Phenotypic Model to Estimate Key Yield related Traits in Field Conditions for UK Bread Wheat

Tahani Alkhudaydi

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

As one of the world's most important cereal crops, wheat is a staple for human nutrition, for example, bread alone provides 20% of UK daily intake [112]. Wheat is grown all over the world on arable land more than any other commercial crops [29]. The increase of population, rapid urbanisation in many developing countries, and fluctuating climate conditions indicate that the global wheat production is expected to have a significant increase in the coming decades [154]. According to the Food & Agriculture Organisation of the United Nations, the world's demand for cereals (for food and animal feed) is expected to reach a billion tonnes by 2050 [1]. Nevertheless, it is critical that this increase of crop production is achieved in a sustainable and resilient way, for example, through deploying new and useful genetic variation [99]. By combining the best genes and traits assembled for target environments, we are likely to increase yield and yield stability to address the approaching global food security challenge [13].

One effective way to breed resilient wheat varieties in fluctuating environmental conditions, which can increase both yield and the sustainability of crop production, is to screen key yield-related phenotypes including the reproductive stage (i.e. flowering time), spikes per unit area, and spikelet number per spike. Based on

the evaluation of these dynamic traits in the field, breeders can select lines based on the yield components, characteristics and hence improve yield potential in finished lines and varieties [133, 58, 85].

The set of protocols and methods which uses plant growth measurements and architectures with a defined accuracy and precision from organs to canopies is called plant phenotyping. Also, it examines how the plant traits are impacted by the genotype and the environment interaction (G x E) [38]. Traditionally, the above procedure was accomplished by specialists using visual inspection of crops and then evaluating key traits based on their personal experience and expertise of the crop, an approach that is labour-intensive, relatively subjective and prone to errors [107, 42].

With rapid advances in remote sensing and Internet of Things (IoT) technologies in recent years, it is technically feasible to collect huge amounts of image- and sensor-based datasets in the field [45, 39]. Using unmanned aerial vehicles (UAVs) or fixed-wing light air-crafts [33, 20, 114], climate sensors [126], ground-based phenotyping vehicles [132, 30], and/or large in-field gantry systems [123, 127], much crop growth and development data can be collected that contains key trait information.

New challenges are emerging from the above data collection approaches including: (1) existing remote sensing systems cannot locate the right plant from hundreds of plots, at the right time; (2) it is not possible to capture high-frequency data to represent dynamic phenological traits in field conditions; (3) our ability to extract meaningful phenotypic information from large sensor and image-based data remains limited; (4) traditional computer vision (CV) and supervised machine learning (ML) are not fit for the sound analysis of datasets because of huge variations in quality and complexity of the in-field phenotyping datasets [14, 38, 119]. Hence, many breeders and crop researchers are still employing the conventional methods of recording, assessment, and selection of lines and traits [84, 60, 16].

The emerging artificial intelligence (AI) based robotic technologies [56, 94, 108]

and distributed real-time crop phenotyping devices [53, 150] have the potential to address the first two challenges as they are capable of acquiring continuous visual representations of crops at key growth stages. Still, the latter two challenges are more analytically oriented and hence require computational resolutions, for example, to segment complicated backgrounds under changeable lighting conditions [2, 98, 153]. CV and ML based phenotypic analysis are therefore becoming more and more popular in recent years.

Most of the examples that utilised CV and ML to extract useful traits in plant research [7, 44, 54, 10, 57, 78, 151, 89, 90, 95, 122] have relied on relatively high quality images, when camera positions are fixed and lighting conditions are stable; however, it is not possible to reproduce image data with similar quality in complicated field conditions, where yield-related traits are assessed. Therefore, in this research, we will explore how deep learning can be used to extract meaningful traits such as detecting spikes and spikelets from infield wheat crop images that are captured under different lighting conditions and at different growth stages and growing seasons.

1.2 Aims and Objectives

In this work, we are going to focus on investigating how deep learning methods can help to extract important key-yield-related phenotypes such as spikelets numbers from uncontrolled infield wheat crop images that are different in lighting conditions at different growth stages and growing seasons. Our objectives are as follows:

- To create and design the required (necessary) datasets for accomplishing the aim of this work, including producing annotations suitable for ML algorithms.
- To investigate whether image pre-processing techniques such as spike segmentation can be beneficial in the context of spikelet counting. To also in-

vestigate other pre-processing techniques such as data augmentation that show good results in image analysis problems.

- To develop deep learning object counting approaches, such as Convolutional Neural Networks (CNN), that are suitable to tackle the problem of spikelet counting in in-field images accurately.

- To investigate the benefits of Transfer Learning using other suitable image sets that may enhance the algorithm performance.

1.3 Research Questions

Our research is trying to solve the problem of quantifying spikelet numbers from wheat crop infield images which differ in sun exposure, environmental conditions, growing stages and growing seasons. The amount of labelled data accessible to us is limited. In this context, we want to adapt a CNN to solve this problem efficiently. In relation to this we ask the following questions:

- Q1: Could eliminating the background from infield wheat crop images simplify the problem of quantifying spikelet counts taking into account the high density of overlapping spikelets and the real environment factors that can add more distortion and complexity to the problem (see Chapter 4).

- Q2: Deep learning models are big data oriented [64, 62] so the more data, the better they can learn. However, with the limited datasets available in the plant phenomics community [121], we ask the following questions in relation to Transfer Learning [83]:

  - Q2.1: Can loading learnt parameters from models trained to solve different tasks on big dataset containing millions of objects benefit our problem of quantifying traits from plants (different task, different data)? (see Chapter 4)

  - Q2.2: Can loading learnt parameters from images of the same plants captured under controlled lab environments be beneficial to solve the same task on limited infield images (same task, different data)? (see Chapter 5)

- Q3: Can Multi-task learning in the context of deep learning be beneficial to our problem specifically and to plant phenomics generally? In other words, can multitask learning tackle the problem of spike segmentation and spikelets counting simultaneously and would this be better than solving those two tasks in sequence? (see Chapter 6)

1.4 Publications

- Tahani Alkhudaydi, Daniel Reynolds, Simon Griffiths, Ji Zhou, and Beatriz De La Iglesia. An exploration of deep-learning based phenotypic analysis to detect spike regions in field conditions for UK bread wheat. *Plant Phenomics*, 2019:7368761, 2019

- Tahani Alkhudaydi, Ji Zhou, and Beatriz de la Iglesia. SpikeletFCN: Counting Spikelets from Infield Wheat Crop Images Using Fully Convolutional Networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 3–13. Springer, 2019

- We aim to publish a journal paper about the work of Chapter 5, which is about counting spikelets and the effect of transferring knowledge from images of similar plants that are captured in controlled environment.

- We aim to publish a journal paper about the work of Chapter 6, which is about the effect of multi-task learning on both learning tasks (spike segmentation and spikelet density estimation).

1.5 Contributions

We have made the following contributions to the field of Plant phenomics using deep learning:

1. We produced a high-quality high-resolution dataset that consists of three sequences : CQ_2015, CQ_2016, and CQ_2017. It represents three growing seasons of four near isogenic lines (NILs) of bread wheat that were cloned at John Innes Centre (JIC) [110, 109]. They were monitored by distributed CropQuant workstations [150] in real field environments and measured manually during the key growth stages in wheat growing seasons from 2015 to 2017. Each sequence is composed of 30 images divided into four key growth stages: booting, heading, flowering and grain filling. Each sequence has two annotated labels : (1) the spike regions in each image have been labelled and (2) each spikelet in each image was annotated by placing a dot in the centre of it. This is presented in Chapter 3 and can be accessed through: `https://github.com/tanh86/SpikeProject`.

2. We conducted a detailed study to show how CNNs [71] can be used successfully to segment and isolate regions of interest (ROI, i.e. spike regions) from noisy backgrounds so that reliable phenotypic analysis could be achieved. In our case the subsequent phenotypic analysis consists of extraction of the spikelet numbers. We have evaluated the models generated for two sequences (CQ_2016 and CQ_17) and for each growth stage. This is presented in Chapter 4.

3. We also conducted a detailed study to show how CNNs [71] can be adapted to perform spikelets density estimation in order to obtain the number of spikelets in an image. We have tested the performance of SpikeCount on three variations of CQ_2016 and CQ_2017: (1) images with full background to measure how noisy background can minimise the model performance (2)

images with the background fully removed to measure how isolating the background helps maximise the extraction of sound phenotyping traits such spikelets numbers (3) finally testing the model on the output of the segmentation model previously implemented to measure the combined performance. This is presented in Chapter 5.

4. We have conducted a detailed study to test the performance of heterogeneous supervised multi task learning model (SpikeMulti) using CNNs to tackle both spike segmentation and spikelets density estimation. Naturally, we have tested the model only on images with background and compare the evaluation with previous results when we follow the conventional pipeline (i.e apply segmentation and the density estimation).This is presented in Chapter 6.

5. We conducted a detailed study to understand the power of Transfer Learning in two ways:

    a) We have employed the Transfer Learning approach using ImageNet images to aid spike segmentation and SpikeMulti [103, 62]. For this, the parameters that were the result of training on a large number of objects that vary in type and domain are used to improve the performance of the learning model, as they may identify suitable features. This is presented in Chapters 4 and 5.

    b) We have used learnt features from wheat images [90] that are in the same domain but captured in a more controlled setting. This allows the convolutional architectures to use the high quality wheat images captured without any noise to provide high quality low level features that can be loaded into convolutional layers and in our case FCN. We have loaded the The Annotated Crop Image Dataset (ACID) [90] parameters into SpikeCount and this helped to improve the results of spikelet density estimation and counting.

6. We have also implemented a multitask algorithm which enables us to get good results from original images with noisy backgrounds and improves on the results of doing the segmentation and spikelet counting separately. This is presented in Chapter 6.

1.6 Thesis Outline

The outline of thesis is as follows. Chapter 2 will discuss the related extended motivation of the topic including the current challenges facing cereal crops and the traditional methods that have been used to face these challenges. Additionally, it will present an overall overview of deep learning and in particular convolutional neural networks (CNN) and object counting techniques. It will explore examples of the tasks that have been achieved in the literature, including by the plant phenomics research community. It will provide a summary of multitask learning with some examples of it used to tackle object counting. Chapter 3 will present the overall proposed methodology for achieving this aim. Chapter 4 will present results from the first step in our proposed approach, that is isolating regions of interest (ROI, i.e. spike regions) by applying semantic segmentation. Chapter 5 will discuss the second step in our proposed approach, i.e. counting spikelets in wheat images as a form of yield quantification for wheat crops, and it will detail the experimental setup and results. Chapter 6 will discuss SpikeMulti, a heterogeneous multi-task fully convolutional model that is trained simultaneously to learn two tasks: (1) pixel-wise classification for spike segmentation and (2) pixel-wise regression for spikelet density estimation used to infer spikelets counts. Chapter 7 will draw insights from the analysis of our results presented in the experimental chapters and will also look at what is left to do in the future.

Chapter 2

Related Work

In Chapter 1 we introduced the motivation and the aims of this work, that is, to study how deep learning can be used to extract important key-yield-related phenotypes such as spikelets numbers from uncontrolled infield wheat crop images that are different in lighting conditions at different growth stages and growing seasons.

The problem is at the intersection of multiple fields that we need to explore and established before selecting the the best approach(es) to tackle it. Therefore, this Chapter will explore the different fields that relate to our research.

In Section 2.1 we provide extended motivation of the topic including the current challenges facing cereal crops, particularly wheat. We discuss current and traditional methods that have been used to face these challenges. We also refer to the shortcomings of these traditional methods and show how the Internet of Things (IoT) has provided some solutions to the flaws of traditional methods. Finally, we list a number of examples of computer vision, machine learning and deep learning used for plant phenotyping.

Section 2.3 establishes a comprehensive overview of deep learning and in particular convolutional neural networks (CNN).

Section 2.4 provides an overview of object counting techniques, different kind of supervisions used and differences between local and global feature extraction. We

explore examples of the task has been achieved in the literature, including by the plant phenotyping research community. We also summarise Multitask learning with some examples of it used to tackle object counting.

2.1 Image Analysis in Plant Phenotyping

Wheat, as presented in the summary of the Wheat Review [111], is considered to be among the top three crops with a total of 607 million tonnes in 2007 according to FAO statistics (http://www.fao.org/faostat/). In addition, it is considered one of the world's most important cereal crops. Wheat is a staple for human nutrition that provides 20% of UK daily intake [112] and is grown all over the world on arable land more than any other commercial crop[29]. The extent of cultivation and diversity is what makes wheat unique and very important in the world today.

Wheat has over 25,000 types and can survive in different conditions of temperatures and environments around the globe. The dominant type of wheat, with over 95%, is bread wheat [111]. The properties of dough made from wheat also contribute to its popularity as it gives many processing choices such as breads, pasta, cakes, etc. These properties are mainly associated with the gluten protein contained within wheat which gives the dough its viscoelasticity, particularly to make leavened bread. In addition, the binding properties of wheat give rise to a wide range of unleavened products such as cakes, pasta and noodles [111].

In terms of health benefits, wheat provides starch, protein and minerals such as Iron and Zinc that could give poor populations many of their essentials nutrients. Shewry [111] says that one of the challenges for the future of wheat is the stability and quality of crops. Climate change and, in particular factors such as temperature and carbon dioxide increases, could potentially affect wheat crop development and its yields. Moreover, lack of access to water, pests and pathogens are considered other negative factors.

Zhou et al. [154] argue that many factors such as population increases, acceleration of growing more urban societies in many developing countries, and changeable climate conditions indicate there will be substantial increase of production of the global wheat in the future where the global demand for cereals is expected to reach a billion tonnes by 2050 [1]. For all those reasons, it is vital to face the increase of crop production through new robust and sustained approaches.

Plant phenotyping as defined by Fiorani et al. [38] as the set of methodologies and protocols used to measure plant growth, architecture, and composition with a certain accuracy and precision at different scales of organisation, from organs to canopies. In addition, it studies how the genotype and the environment interact (G x E) to impact plant traits.

Reynolds and Langridge [99] explain that there are three important factors to crop improvement : (1) leveraging the knowledge of physiological research to design new models that improve the plant processes and crop ideotypes, (2) evaluating complex trait expressions using high throughput phenotyping technologies and applying them realistically on a breeding scale in field environments, and (3) deploying new and useful genetic variations. To address the imminent challenges of global food security we can combine the best genes and traits assembled for target environments, which likely lead to increased yield and yield stability [13].

One effective way to breed robust wheat varieties in changeable environmental conditions, which can increase both yield and the stability of crop production, is to screen key yield-related phenotypes including the reproductive stage (i.e. flowering time), spikes per unit area, and spikelet number per spike. For example, pask et al. [85] discuss 20 measurements of wheat traits. Yield and yield components measure the production of grains which is the ultimate expression of all physiological processes. Throughout the crop growth cycle, these processes have interacted with the weather and environment and there is an urgent need to accurately measure these traits to exhibit the relationship between physiological characteristics and productivity. These trait measurements are spike (ear) number (SNO), grain num-

ber (GNO), grains per spike (ear) and thousand grain weight. The measurements can be done through crop sampling and in-field surveillance [85]. Based on the evaluation of these dynamic traits in the field, breeders can select lines according to their yield components and therefore improve yield potential in finished lines and varieties [133, 58, 85]. The above procedure, conventionally, was achieved by specialists using visual inspection of crops and then assessing key traits based on their personal experience and expertise of the crop, an approach that is labour-intensive, relatively subjective and prone to errors [107, 42].

In recent years, the accelerated advancements in both Internet of Things (IoT) and remote sensing have made it technically attainable to collect vast amounts of image- and sensor-based datasets in the field [45, 39]. Many crop growth and development data can be collected that contains key yield-related traits through different types of remote sensing.

For example, low-altitude unmanned aerial vehicles (UAVs) are used to estimate plot-level ground cover for cotton, sorghum and sugarcane to extract ground cover, an important physiological traits [33]. Also, fixed-wing light air-crafts are used to extract ground cover in sorghum, canopy temperature in sugarcane and three-dimensional measures of crop lodging in wheat [20], and to develop prototype crop information systems that are based on remote imaging sensor, e.g. Moderate Resolution Imaging Spectroradiometer (MODIS) to monitor opium and cereal crops [114]. Another example is climate sensors which involve collecting heterogeneous environmental information such as temperature, solar radiation, humidity, pH, moisture and wind. Then, all collected information is merged to generate a response for every scenario [126]. Ground-based phenotyping vehicles are equipped with multiple sensors such as LiDAR, RGB camera, thermal infra-red camera and imaging spectroradiometer to records crop data on multiple plots during crop life cycle and give the opportunity to measure and extract traits several times [132, 30]. Finally, large in-field gantry systems such as LeasyScan [123] a novel 3D scanning mechanism to monitor canopy traits related to water usage. Another example is

Field Scanalyzer, a fully automated field phenotyping platform used to provide a consistent observation and monitoring of crop growing and generate detailed information of canopy through crop life-cycle [127].

There are emerging challenges for the previously explained data collection methods : (1) current remote sensing systems limitation on locating accurately the right plant from hundreds of plots, at the right time; (2) it is not possible to capture high-frequency data (e.g. resolution of minutes to hours) to represent dynamic phenological traits (e.g. booting and anther extrusion) in field conditions; (3) limitations in extracting meaningful phenotypic information from large sensor- and image-based data; (4) traditional computer vision (CV) and supervised machine learning (ML) are not fit for the robust analysis of datasets because of huge variations in quality and complexity of the in-field phenotyping datasets (e.g. high-dimensional multi-spectral imagery)[14, 38, 119]. Therefore, many breeders and crop researchers are still employing the conventional methods of recording, assessment, and selection of lines and traits [84, 60, 16].

The rising artificial intelligence (AI)-based robotic technologies [56, 94, 108] and distributed real-time crop phenotyping devices [53, 150] have the potential to address the first two challenges as they are capable of acquiring continuous visual representations of crops at key growth stages. Still, the latter two challenges are more analytically oriented and thus require computational resolutions, for example, to segment complicated backgrounds under changeable lighting conditions[2, 98, 153]. CV and ML based phenotypic analysis are therefore becoming more and more popular in recent years.

There are many examples of using CV and ML to extract plant traits. LemnaGrid [7] utilises colour and intensity features to segment Arabidopsis leaves from predefined background. PhenoPhyte [44] employs the OpenCV[54] library to semi-automate object segmentation based on colour space and adaptive thresholding, so that leaf phenotypes can be measured. MorphoLeaf[10], a plug-in of the Free-D analysis software, performs morphological analysis of plants to study different plant

architectures; BIVcolor[57] uses a one-class classification framework to determine grapevine berry size using the MATLAB's Image Processing Toolbox. Phenotiki [78] combines off-the-shelf hardware components and easy-to-use Matlab-based ML packages to segment and measure rosette-shaped plants.

Leaf-GP [151] combines Python-based image analysis libraries (e.g. Scikit-Image [124] and OpenCV) and the Scikit-Learn[86] ML library to measure growth phenotypes of Arabidopsis and wheat based on combining colour, contrast, and morphological features.

Root Estimator for Shovelomics Traits (REST) software [28] analyses root stocks sampled using novel splitting methods which are then photographed under controlled illumination. This was used to develop precise and high throughout software which was evaluated on 33 diverse maize hybrids. Vasseur et al. [125] developed ImageJ [93] macros to extract shape descriptor rosettes (from tray or individual pot images) or to perform segmentation (from inflorescence images). This was used in a predictive statistical model using R to predict plant dry mass and fruit numbers. Image Harvest (IH) [59] is an open-source analysis framework which can be used to carry out a parallel processing of images generated from multiple high-throughput plant phenotyping platforms. It provides many selections to extract traits from images. EasyPCC [46] is a tool provided to evaluate the plant canopy coverage (PCC) from a large number of images with different field conditions using pixel-based segmentation (decision trees). BioLeaf [74] is a free mobile application that measures the loss of foliar in soybean leaves based on two techniques: Otsu segmentation and Bezier curves. saRIA [80] is a GUI-based semi-automated root image analysis tool that applies image segmentation using adaptive thresholding and morphological filtering.

State-of-the-art deep learning (e.g. Convolutional Neural Network, CNN) has been employed to carry out indoor phenotyping for wheat root and shoot images using edge- and corner-based features[89]. A homogeneous multitask deep learning model is developed to count and localise wheat spikes and spikelets from wheat crop

images captured in a controlled environment inside a glasshouse [90]. Finally, significant advances have been made in the deep learning approaches to automate leaf object segmentation and phenotypic analysis[95, 122].

2.2 Internet of Things (IoT) in Plant Phenotyping

According to Madakam et al. [75] the best definition of IoT is : "An open and comprehensive network of intelligent objects that have the capacity to auto-organise, share information, data and resources, reacting and acting in face of situations and changes in the environment". The advances of technologies such as hardware improvements (the reduced size and power consumption), connectivity improvements (wireless connections) and cloud computing have allowed to employ IoT into building a network of devices that share data and information [81, 148]. There are four layers presented on the proposed IoT agriculture platform [22, 105, 81]: application, processing, transport and perception which can be related to the main components of any IoT system: application, services, network and devices.

The perception layer consists of the physical devices in the platform and how they interchange with each other and with the transport layers which collect data to allow the communication of information. UAV devices, sensor nodes or devices developed for specific purposes using components like sensors and single-board computers (SBC)—such as Arduino or Raspberry Pi [81] are used. Sensor nodes have been used, for example, to detect and monitor leaf disease using integrated sensor networks to measure moisture, temperature and humidity which are utilised in various locations around the farm and controlled using one controller called Raspberry PI (RPI) [120]. An example of interaction between the perception layer and the service layer using the transport layer is SmartFarmNet [55]. SmartFarmNet is an IoT-based platform that automatically collects environmental, soil, fertilisation, and irrigation data. Then, it automatically correlates these data to assess the crop performance. It provides a virtual integration of any IoT device such as

sensors, cameras, weather stations and it stores their generated data in the cloud for analysis.

The transport layer relates to the network and transport abilities, such as network and application protocols [22, 81]. A network protocol in any IoT system is there to facilitate the communication between the perception layer and the processing layer. The network protocols are divided according to data exchange into short-range, cellular networks and long-range, taking into account the power consumption and exchange rate. For example, a low-cost long-range protocol was presented for automatic cloud-based irrigation [35].

The processing layer consists of data storage, visualisation and processing resources which enable distributed storage and parallel data processing and lead to extracting the required information quickly. That information can be fed into artificial intelligence (AI) systems to detect patterns and correlate complex data [81].

The application layer provides the farmers with management information to empowers them to make well rounded decisions regarding crop management.

2.3 Deep Learning

For many years machine learning has helped to solve image recognition tasks. Supervised and unsupervised learning are considered the main themes in machine learning. In the supervised approach, iterative training is conducted using previously labelled data to learn a certain mapping from the samples features to either discrete (classification) or continuous (regression) output. For example, when a classifier is trained to detect a certain object in an image, a label must be associated with the image to indicate whether it is the target object or not. Unlike supervised learning, unsupervised approaches do not require data labelling. Clustering is the main unsupervised task that groups the data points into clusters based on their features, and in particular by using some concept of similarity. For example, Haralick et al. [48] have approached image segmentation as a clustering task such as

applying k-means to cluster image pixels [49]. As a compromise, semi-supervised learning has emerged where the learning process uses labelled and unlabelled data to produce the mapping [19].

Traditionally, when approaching any image recognition task using machine learning, an image is transformed into handcrafted features and they are chosen based on the expertise and knowledge of the researcher and to the real world problem under consideration. Afterwards, the handcrafted features are manipulated using supervised or unsupervised learning according to the task.

2.3.1 Shallow Vs Deep Neural Networks

The Shallow Neural Network is based on the principle of a Perceptron [101]. A *Perceptron* is a linear function that maps a weighted sum of the inputs to either 1, if the outcome is larger than or equal to a certain threshold θ or 0, otherwise (see Equation 2.1):

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq \theta; \\ 0 & otherwise. \end{cases} \tag{2.1}$$

Minsky et al. [79] explained the disadvantage for solving functions where their inputs are not linearly separable such as XOR function. In other words, the perception is only capable of solving input data that can be separated with a linear decision boundary e.g AND operation. Thus, *multi-layer perceptrons (MLP)* have emerged to solve this limitation by stacking intermediate layers of perceptions, called hidden layers, between the input and output layers.

One of the MLP properties is the capability of representing any function with only two hidden layers because of the fact that any function can be approximated by a number of simple surfaces. To approximate each surface, a neuron can be added to the second hidden layer. Therefore, the more complex the function, the more surfaces are needed to approximate it and, as a consequence, the more neurons are

added to the hidden layers. This causes the size of the MLP to grow exponentially [128].

On the other hand, deep networks have the capacity to learn complex representation by adding more layers than adding neurons in each layer. Delalleau et al. [31] compared the number of neurons used to represent a certain function in terms of the size of the input vector n. They found that for a network with depth i (number of layers), the overall number of neurons used is $O(n)$. However, in two-layer shallow networks, the number of neurons used is $O((n-1)^i)$. Many works have been reviewed in [128] that justify the popularity of deep nets over shallow networks. Back-propagation (BP) [102] is widely used as an algorithm for learning parameters in deep networks.

2.3.2 Convolutional Neural Nets (CNN)

Convolutional neural nets are fundamentally inspired by the visual vortex in humans. Any CNN model has three main layers: convolutional, non-linearity and subsampling layers. The convolutional layer is based on the convolution operation. This operation transforms any image into another image by multiplying it with a kernel or a filter. Many different filters can be applied to extract different features. The benefit we get from applying a convolution operation is detecting local features that are invariant to translation transformation. Usually, the kernel or filter matrix is predefined and fixed. However, in the convolutional layer, these filters are considered parameters and we train the CNN to learn them. This gives us a powerful advantage because the model learns only the kernels that are relevant to the given images and problem domain. Typically, the second layer is composed of the non-linearity units which strengthen the convolution operation result by highlighting the features that are useful for the prediction task. *Rectified Linear Unit (ReLU)* [62] is used widely because it decreases the training time of deep nets compared to other non-linearity functions (Figure 2.2). It is a threshold function (see Equation 2.2), which is an element-wise (x is a feature map element) thresholding operation

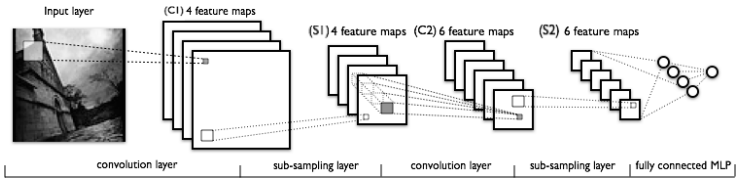

Figure 2.1: A LeNet Architecture. LeNet [65] is the first CNN architecture, it has two groups of convolutional layer and subsampling layer.

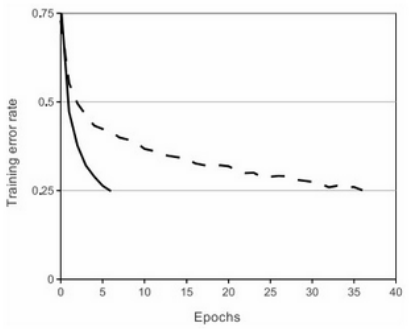

Figure 2.2: ReLU Performance [62]. The solid learning curve of CNN ImageNet classier that uses ReLU,shows how the learning rate is faster than the same CNN model which uses other non-linearity function.

that is applied to the outcome of the convolutional layer (resulted feature map) to suppress negative values:

$$f(x) = max(0, x) \tag{2.2}$$

where x is an element in the feature map.

The third essential layer is subsampling and its role is to sample some values from a certain region or window of the input. The size of the window is predetermined when setting up the model. There are two types of subsampling: *max-pooling* which is to sample the maximum values and *average-pooling* which is to sample by averaging the input values. The goal of subsampling is to ensure that the features are invariant to any noise or transformation, also to down-size the feature map to do less computation. To some extent this step mimics extracting local features from the image which are more robust to object variations or noisy background.

In principle, any CNN would have these three layers with a certain size and number; then, typically, another three layers with a different size and number. The

advantage of this interchangeable order is finding more robust features. To perform the classification or regression task, two important layers are connected at the end the model: a fully-connected layer and output layer.

2.4 Object Counting

Object counting from images is a problem that emerges in many different scenarios, for example, monitoring crowds [92, 18, 143], performing wildlife census [6], counting blood cells in images [47] and others. Sometimes, the problem is framed in the context of counting objects in still images [5, 21, 6] whereas in other cases, the problem involves counting moving objects in videos [40]. The latter has the additional complication of not only detecting objects but also tracking them as they move in the frame, for example pedestrian tracking [129, 130]. Supervised counting methods include a number of main themes: counting by global regression, detection, density estimation, segmentation, and subitising. Unsupervised approaches have been used for object counting but according to Lempitsky and Zisserman [66] they tend to produce less accuracy. To reduce the burden of labelling objects semi-supervised approaches [18] have also been proposed in which only some data is labelled and unlabelled data or labelled data from other scenes is used to improve the model.

2.4.1 Full vs Point-level Supervision

Often machine learning approaches to object counting are supervised approaches, i.e. they require some labelling of the images so that there is some ground truth about the number of objects. Labour intensive annotations such as bounding-box and pixel-wise annotations are rarely used and do not mimic the way humans count by pointing at objects [66]. Dotting annotation has been introduced and used vastly in the object counting area [5, 6, 37, 66, 143]. There are two types of dot annotation in the literature: single and crowdsourcing. The first type is done by a single

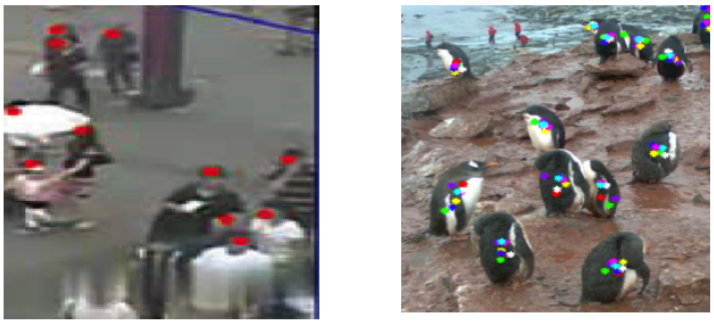

Figure 2.3: Examples of single dot annotation [143] (left) and crowdsourcing dot annotation (right) [6] The example on the *right* [6] shows a result of an image that were dot annotated by crowdsourcing where multiple annotators put a dot on each object. Each annotator has a unique colour where the example on the *left* [143] shows a single dot annotation.

annotator whereas the second (crowdsourcing) dot annotate the same data multiple times. Arteta et al. [6] argue that there is a negative correlation between annotation simplicity and model complexity, that is, the simpler the annotation the more complex the model will be. Therefore, while it is true that dot annotation lowers the burden of annotation, it requires more cues regarding the spatial information of the objects [6, 143]. Crowdsourcing dot annotation can provide a solution to fill this gap by exploiting the point patterns, which can provide spatial cues about the objects and be used for any complex task such as counting [6].

2.4.2 Global vs Local Features Extraction

According to Seguí et al. [106] a counting system is a mapping from the image space to a set of non-negative integers. This requires the definition of the image features and also of a mapping function. Traditional approaches have relied on handcrafted features. For example, features may relate directly to the objects being counted, to the background or to groups of objects. There are two types of hand crafted features in terms of object counting: global features [25, 61, 43] and local features [37, 73, 104]. Global features are dependent on the values (pixels) of the entire image [139] and are mostly shape and texture descriptors. However, they are

occlusion and clutter sensitive [69]. Cho et al. [25] extracted three global features among all images: length of edges of the crowd objects, the crowd objects density and the background objects' density. Kong et al. [61] used edge orientation and blob size histograms, which were generated from edge detection and background subtraction, to estimate crowd density. Giuffrida et el.[43] used an unsupervised feature extractor to build global features to count leaves in rosette plants.

On the other hand, local features are dependent on the values of a certain neighbourhood, a patch or region of the image [139]. Local features give the advantage of recognising the object even if the scene is cluttered or occluded [69]. Fiaschi et al. [37] employ random forest regression to map local features such as dense features (features generated at pixel-level) to the density estimates. Ma et al. [73] proposed a local texture feature mapping: gradient orientation co-occurrence matrix(GOCM), where they argue that they are more discriminative and representative of the crowd patches than a co-occurrence matrix (GLCM). Ryan et al. [104] argued that global features require an enormous training set because of the variety in crowds. Therefore, they proposed using local features that are related to individuals and small groups.

As for the mapping function, machine learning can deliver a number of different approaches either in the form of multi-classifiers, if we have samples available for all possible cardinalities or, more commonly, multivariate regression approaches, if we want to map from the feature space to a set of (bounded) positive integers.

2.4.3 Object Counting Methods

2.4.3.1 Counting by detection

Training an object detector requires us to provide the system with sufficient labelled and localised objects to represent the different appearances of the object. Once object detection is achieved, counting is relatively trivial. However, detection could be challenging particularly in overlapping objects causing occlusion, differences

in scale, and also a lack of detectable background between distinct objects [6]. Detection of humans is often used in crowd counting [134, 68] but humans can adopt many different postures which may make human body recognition more challenging.

2.4.3.2 Counting by segmentation

Many works [17, 104, 40, 6, 95] have utilised segmentation in various ways. For instance, Chan et al. [17] used foreground segmentation to segment the crowd into homogeneous groups based on the crowd direction using dynamic texture segmentation. On a segment level, a number of features are extracted such as segment area, perimeter, perimeter edge orientation and perimeter area ratio. Globally, internal edge features were computed on the entire image and then masked by the segmentation. Also, each image patch was classified in terms of crowd density using grey-level co-occurrence matrix (GLCM) to extract texture features. After that, using regression, the number of people in each group will be estimated based on each segment's features. On the other hand, Ryan et al.[104] followed the same approach in [17]. They extracted the same features: area, perimeter, perimeter-area ratio, edges, and edge angle histogram but after applying crowd segmentation. Therefore, they utilise regression on only locally extracted features.

2.4.3.3 Counting by regression

When the only task required is to determine the total number of a certain object in an image rather than detecting it, counting by regression may be easier due to the difficulty of the detection problem. Also, it may be the only way if the image suffers from severe object overlapping or if the resolution of the image is low. Counting by regression often maps image global and/or local features to the object total number from the whole scene or various extracted parts of the scene.

For instance, as discussed in the counting by segmentation methods, Chan et al. [17] utilised counting by regression as the final step to count the number of people in each group based on each segment's features extracted from each image patch. Also, after applying crowd segmentation, Ryan et al. [104] applied counting by regression based on the features extracted from each segments to estimate the count [104].

Chen et al. [23] proposed a model that learns a functional mapping from low-level features to multi-dimensional structured outputs to predict the count of pedestrians. Those low-level features consist of three parts. First, segment-based features: foreground area, total number of foreground perimeter pixels and the ratio of perimeter area; also, structural-based features such as edge pixels total number and edge orientation histogram; lastly, local texture features such as Gray Level Co-occurrence Matrix (GLCM).

2.4.3.4 Counting by density estimation

Unlike other counting regression methods that use global region features, learning to count objects through density estimation regression [66] takes into account the spatial information of objects. This, to some extent, would give localisation information regarding the object occurrences in the image. Density estimation regression learns mapping from local features into pixel level densities. This gives the advantage of integral density estimation over any image regions. In other words, it can estimate the density of any region in the image. Lempitsky and Zisserman [66] used dot annotations to infer density maps and utilised them as training ground truths by applying a normalised 2D Gaussian kernel. Then, they designed a counting cost function that minimises the distance between the target density map the inferred ground truth one. Subsequently, Fiaschi et al. [37] used random forest regression, which optimised the training process to predict the density map. Exploiting the advantage of integral density estimation over any sub region of the image, a counting system was presented that estimates object counts interactively [5]. Pham et

al. [87] proposed a patch-based method to solve crowd density estimation exploring structured learning using random decision forests. A fast density estimation based visual object counting (DE-VOC) was presented which leveraged the mapping between images and their corresponding density maps in two different features spaces [131]. Xu et al. [136] proposed that using more image features can lead to more robust crowd density estimation models. Also, they utilised more efficient random projection forests to reduce the dimensionality resulting from traditional random forests.

2.4.3.5 Counting using deep learning algorithms

Convolutional Neural Networks (CNN), which are capable of learning their own features, are beginning to show real promise in the area of object counting [6, 21, 143]. Seguí et al. [106] argue that it is possible to learn to count using CNN focusing on the multiplicity of the object of interest, without annotating the objects via detection or bounding boxes. They achieve this in their CNN by using a two block structure, with the first block made up of two or more convolutional layers which tries to capture and represent the concept they are counting and a second block of fully connected layers that produces the counts.

Arteta et al.[6] used foreground segmentation to aid the learning of density estimation. They trained a segmentation map and a density map jointly to count penguins. French et al. [40] used segmentation to count fish in a two stage process:(1) foreground segmentation and (2) CNN edge detector to accurately segment the severely occluded fish areas.

A recent work by Ren and Zeme [95] has used deep learning algorithms such as Recurrent Neural Nets(RNNs) and CNN to employ visual attention to produce instance segmentation while counting. They adopt visual attention that humans use when counting. They locate and segment the object iteratively one by one and keep track of them. They claim that this approach is more efficient in crowded

scenes than the regression based model that utilises global features.

Chattopadhyay et al. [21] explore applying deep learning to count different types of objects in everyday scenes, and propose to solve it using an approach based on developmental psychology. Their method, **subitising**, is based on the ability of humans, and even children, to map a perceptual sign to a numeric estimate. They argue that everyday scenes differ from some of the scenes used in this area of research because many different objects can appear, some with zero counts, some with multiple counts. They solve the counting problem by dividing it into subproblems on different cells in a non-overlapping grid and then adding the partial counts. They also take care of context across cells to improve their count. However, they require annotated scenes as input with object bounding boxes and they also require category wise counts for each type of object.

Another important problem in object counting that emerges in crowd counting problems is that small changes to the scene may make the learned models useless for new data as models are often scene specific (e.g. [129, 143]). For example, changes in lighting, resolution, distribution of crowds, and perspective can make models poor in new scenes. It is important to develop models that can work across different scenes as such models may have a much wider applicability. Scene specific models may be more accurate, but need to be re-trained when applied to a new scene and that may involve labelling the new scene, which if done manually may make the approach costly and not scalable. There have been a number of attempts to solve the problem of scale variation and perspective distortion [6, 143]. For example, one proposed solution is to use a perspective map produced by selecting random pedestrians and labelling them from head to toe [143] taking into account that the average adult height is 175 cm. One pixel in the resulting map equals to one meter of pixels in the original image at the same location. On the other hand, Wang and Wang [129] propose an approach in which the generic model is adapted to a specific scene by selecting examples from the scene that may help with re-training.

2.4.3.6 Object Counting in plant phenotyping

Counting constituent parts of plants is an essential and important task to be tackled in plant phenotyping. For example, TasselNet [72] was developed to count maize tassels from infield maize crop images captured from 2010 and 2015 in various fields in China. TasselNet performs counting by local regression using a deep convolutional neural network-based approach. It was tested on 8 test sequences achieving good results.

Another example of counting in plant phenotyping closer to our own application is counting spikes and spikelets in wheat. Spikes and spikelets represent important yield traits for wheat [67]. In this context, Pound et al. [90] developed a multi-task deep learning model to count and localise wheat spikes and spikelets, achieving an accuracy of 95.91% and 99.66% respectively using non-maximal suppression (NMS). They tested the model on wheat crop images captured in a controlled environment inside a glasshouse. Their problem is therefore similar to ours but simpler given the reduced variation in the controlled laboratory environment as opposed to a real field image.

Also, Madec et al. [76] investigated counting spikes from infield wheat crop images captured by UAV platform using two CNN-based models. The first was Faster-R-CNN [96], a CNN based object detection model. The second was an adaptation of TasselNet [72] for this task, the CNN-based local regression model developed to count maize tassels from infield images. They concluded that both models achieved similar results when tested on images containing crops that have a similar distribution of spikes as the images both models trained on. However, they found that Faster-RCNN outperformed other models when tested on images containing more mature crops.

Fernandez-Gallego et al. [36] developed an image processing pipeline to count and localise wheat spikes (also known as ears) by segmentation. They tested their pipeline on infield durum wheat crops with two growth stages and reported high

success rate (over 90%) between algorithm count and manual (image based) counts.

Also, Hasan et al. [50] tackled the problem of counting spikes by using an R-CNN object detector. They trained four versions of R-CNN on four different growth stages of infield wheat images that vary in growth stage and variety. They again reported good results with detection accuracy ranging from 88 to 94% across different sets of test images.

2.4.4 Multitask Learning (MTL)

Multitask Learning (MTL) is a branch of machine learning, where multiple related targets are trained simultaneously to solve complex goals [15]. Caruana [15] argue that when it comes to solving complex goals, relying on targets modularity and solving each one independently can be pointless and ineffective. In the context of object counting, Zhang et al. [143] developed a learning algorithm with multiple learning objectives which are switchable. Their objectives are crowd density and crowd counting and they can help each other to obtain better estimates. Another example of multitask learning for counting is the method of Arteta et al. [6] for counting penguins. They introduced a multitask convolutional Neural Net architecture that primarily learns a segmentation mask to help the object density map regression. More specifically, the model is trained jointly to learn three targets: a segmentation map, a density map and a uncertainty map. The segmentation map uses a pixel-wise binary classification approach with a hinge loss. The remaining targets are regressed with a root-squared loss. Zhao et al. [149] proposed two learning phases to solve the crossing line people counting problem. For instance, in surveillance videos, they count the number of people across a line-of-interest (LOI). First, they employed CNN to learn to predict two targets, the crowd density map and crowd velocity. Then, they trained a CNN to learn a crowd counting map. Sindagi et al. [116] proposed to train CNN jointly to learn crowd count classification and density map estimation to tackle crowd counting, which they argued, lowered the count error and improved the density maps of highly challenging datasets.

2.5 Summary

In this Chapter, we have extended the motivation behind the main aim of this thesis: employing deep learning to extract key-yield phenotypes such as spikelet numbers from uncontrolled infield wheat crop images. We produced a detailed explanation of the challenges that currently face cereal crop production such as wheat crops. Also the limitations of conventional phenotyping methods have been discussed. In addition, we explained the importance of the IoT and various data collection mechanisms that help with some of the limitation of traditional phenotyping methods. However those generate new challenges of their own. We also showed that most of the work employing CV and ML in plant phenotyping has been done based on plant images that are captured in relatively controlled settings and consistent lighting. Those do not resemble the complexity and variability of real infield crops. In addition, IoT can collect massive dimensional data. We have given a comprehensive review of deep learning and have shown that it can be successful in extraction of meaningful information related to crop development.

In relation to object counting, we listed the major object counting methods used in the literature with some examples for each one. We believe that the problem of spikelet counting from infield wheat images is similar in many aspects to crowd counting such as variability, self-similarity, varying backgrounds, severe occlusion, and density. The problem of density is widely tackled using the methods of density estimation which we discussed. In addition, we explained how the multitask learning approach is used in the context of counting objects, and how it can be beneficial in solving the problem in hand.

In the next Chapter, we are going to describe the overall proposed methods and the motivation behind selecting them. Also, we are going to introduce the dataset used to test and evaluate the selected methods.

Chapter 3

Methods

Figure 3.1 shows the type of image we are going to analyse, together with a close up of spikelets with their spikes. Our objective is to count the spikelets as accurately as possible in such uncontrolled infield wheat crop images. For this, we employ deep learning methods to extract spikelet numbers. In this Chapter we are going to present the overall proposed methodology for achieving this aim.

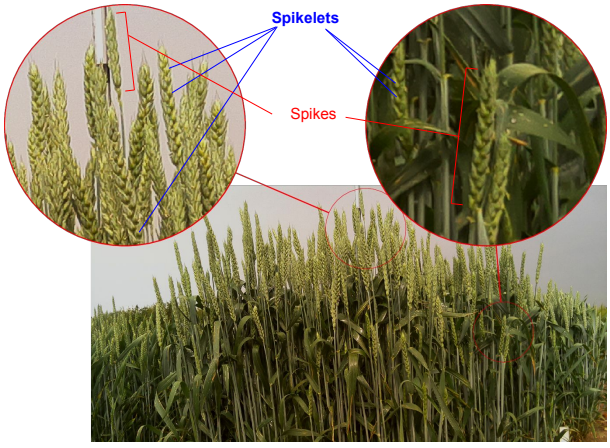

Figure 3.1: An example of wheat crop scene, spikes, and spikelets

There are five components to be introduced within our proposed methodology:

1. We start with a formal definition of our problem. We then determine which object counting method is most suitable for our problem and the reason(s) behind selecting this methods. We discuss how to model the problem using the selected object counting method.

2. We explain the workflow we follow along with with a description of learning tasks used in this workflow.

3. We present a detailed description of the datasets we used for testing and evaluation. Also, we present the protocol for training and validation, explaining how data preparation is carried out for training and validating our approach.

4. We discuss which deep learning networks are more suitable to accomplish the learning tasks.

5. We give a general description of how we utilise Transfer Learning in the different learning tasks.

6. In the final section, we describe the testing and evaluation of the proposed methods detailing the quantitative metrics for each learning task.

3.1 Proposed Methodology

3.1.1 Problem Statement

In Chapter 2 we explored the different object counting methods. Counting by segmentation and detection, which involve segmenting or detecting the objects individually to be counted. This in the context of counting spikelets is unrealistic due to the severe density of occlusion. Therefore, we model the problem in hand as a crowd counting problem [117] which shares similar features such as severe occlusion, high number of objects etc. Most crowd counting work has utilised initially global regression methods. However, this relies on regressing the global count and ignores the spatial information which could lead to more accurate counting. [117]. Density estimation, which preserves the spatial information, has been leading to better results in the context of crowd counting. It is a well-known technique that originally learns a linear mapping from the image to a density map [66].

Figure 3.2: An example of spikelets density generation where: (a) represents sub-image of a wheat crop, (b) represents corresponding dot annotation and (c) the generated density map from the dot annotation

Next, we define the problem of spikelet counting mathematically. We model it as a density estimation problem as follows. Given N input images $I_1, I_2, ..., I_N$ with a size of $H \times W \times D$, in our cases representing infield wheat crop plots, for each image, I_i, there is a corresponding dot map P_i that can be represented as a set of 2D points $P_i = \{P_1, ..., P_{SPC_i}\}$, where $|P_i|$ is the number of spikelets in image I_i. Each point is placed at the centre of each spikelet as shown in Fig 3.2(b). To generate the ground truth map GT_i (shown in Fig 3.2 (c)), a 2D Gaussian kernel $\mathcal{N}(p; P, \sigma^2 1_{2\times 2})$ is applied to the dot map P_i which generates a density for each pixel p of image I_i. Therefore, the size of GT_i is the same as the input image: $GT_i = \{D^{P_1}, ..., D^{P_{H\times W}}\}$ where D^{P_j} is the generated density for the j^{th} pixel in image I_i.

The effect of applying the Gaussian kernel is that it can reflect the crowding around a spikelet by taking into account the information of the pixel's neighbourhood when updating its density value. In other words, the more spikelet occlusion in a certain region, the higher the density values will be assigned to pixels in the region.

The total number of spikelets in a certain image I_i is the sum of all pixels densities in density map: $|P_i| = SPC_i = \sum_{p \in I_i} D^p$.

3.1.2 The outline of proposed workflow

The complexity of the problem is enhanced by noisy background objects which could be an obstacle to extracting spikelet numbers accurately. One important

step before extracting plant growth traits from images is to isolate the ROI from noisy and redundant background [70]. Figure 3.3 illustrates the overall proposed workflow we are going to follow is composed of two learning tasks:

1. Apply foreground semantic segmentation using deep learning to isolate ROI, in our case spike regions, which in themselves are important traits to extract; we will refer to this model as (SpikeSEG).

2. Apply a deep learning algorithm to perform density estimation and extract the number of spikelets from the segmented images (the output from step 1), we will refer to this model as (SpikeCount).

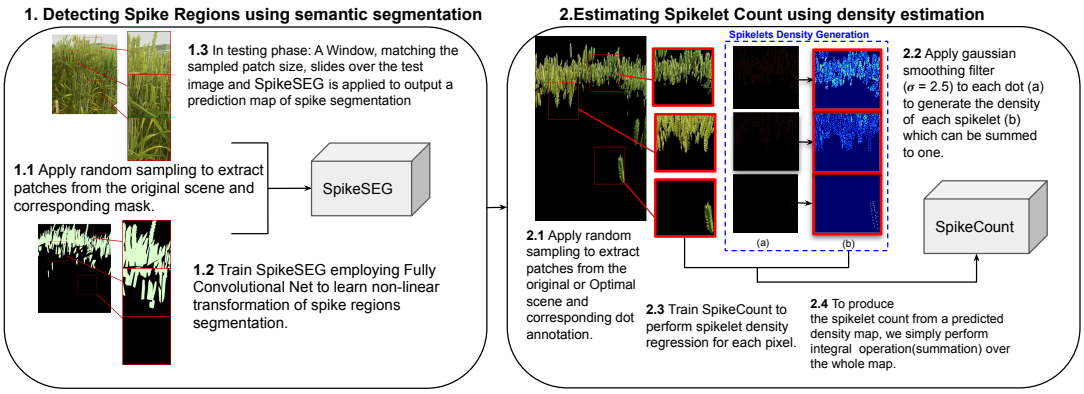

Figure 3.3: The overall proposed workflow illustrates the first step, detecting spike regions using semantic segmentation and the second step of estimating spikelet count using density estimation. It also details the training, validation and testing strategy for developing SpikeSEG and SpikeCount models

3.1.3 Fully convolutional network (FCN)

An important part of our methodology is to determine which deep learning network is most suitable for our learning tasks. We propose to employ a FCN [71] to tackle both semantic spike segmentation task (SpikeSEG) and spikelet density estimation (SpikeCount), because both tasks are fine grained and require pixel-level prediction. Also FCN and varieties have widely been employed to tackle density estimation [6, 145, 77, 11]. For those reasons, we decided to select FCN to tackle both tasks as

this simplifies the experimental setup and enables us to use the same to investigate multi-task learning (SpikeMulti) (Chapter 6). However, we will discuss other deep learning models that can be used for segmentation in Chapter 7 as part of our further work.

The novelty and advantage of applying FCN in this context is that it transforms the non-spatial output produced by the deep classifier to a spatial one that is required for both tasks. This is accomplished through transforming the fully connected layers attached at the end of the deep classifier, so that image level prediction can be produced. Fully convolutional layers can preserve the spatial information of target objects and hence enable the pixel level prediction. This approach provides a solution to localise and detect targeted objects based on manually labelled training datasets. However, the output of the FCN at this stage has lower resolution than the original input image and yields a coarse output. To tackle this down sampling problem, FCNs were proposed to reverse the effect of repetitive sub-sampling through up-sampling [115]. The up-sampling method is based on backwards convolution (also called deconvolution). Furthermore, a FCN provides another enhancement by applying a concept called the skip technique. This takes advantage of the hierarchy resulting from any convolutional neural network that starts with local feature maps describing the finest information (i.e. edges, contrast, etc.) and ends with the coarsest information that describe the semantics of the target objects (i.e. the more generic features of the region). A FCN combines those levels to produce a more detailed output map. Those two enhancements can lead, in the case of density estimation, into a more accurate density map and hence accurate spikelet numbers.

3.1.4 Datasets

Next, it is important to present the datasets used in this research, how they were collected, and what is the general protocol for evaluating different methods. Lastly,

we explain in general the proposed methods to solve the problem and finally the evaluation metrics for each method.

3.1.4.1 Wheat field experiments

To assess key yield-related traits for UK bread wheat, we have utilised four near isogenic lines (NILs) of bread wheat in field experiments, representing genetic and phenotypic variation with the similar genetic background called "Paragon", an elite UK spring wheat that is also used in the Biotechnology and Biological Sciences Research Council's (BBSRC) Designing Future Wheat (DFW) Programme. The four NILs include Paragon wildtype (WT), Ppd (photoperiod insensitive) and Reduced Height (Rht) genes (semi-dwarf) genotypes cloned at John Innes Centre (JIC)[110, 109], which were monitored by distributed CQ workstations in real field environments and measured manually during the key growth stages in wheat growing seasons from 2015 to 2017.

3.1.4.2 Image acquisition

The high-quality Red-Green-Blue (RGB) image series used in this study were collected from 1.5-metre-wide (5-metre-long) wheat plots during a three-year field experiment. To generate continuous vision representation of key growth stages of the crop in the field, four CQ workstations were dedicated to conduct high-frequency (one image per hour) and high-resolution (2592x1944 pixels) imaging in order to acquire target yield-related traits expression. Each CQ workstation or terminal consists of a single-board computer, climate sensors, a sensors integration circuit, an imaging sensor, a local storage, network components, Debian operating system, and custom-made Python software package for crop images and climate data collection [152]. Between May and July in three growing seasons (i.e. covering booting, GS41 – GS49, to grain filling stages, GS71 – GS77), over 60 GB image datasets

have been generated by CQ devices. For each growing season, 30 representative images were selected for the deep-learning based phenotypic analysis.

In order to maintain similar contrast and clarity of wheat images in varied lighting conditions in the field, the latest versions of open-source picamera imaging library[88] and Scikit-image[124] were employed to automate the adjustment of white balance, exposure mode, shutter speed and calibration during the image acquisition [152]. In-field image datasets were synchronised with centralised storage at Norwich Research Park (NRP) using the Internet-of-Things based CropSight system [97]. Figure 3.4 shows examples of the wheat plot images acquired by CQ workstations from 2015 to 2017 (in columns), indicating that images selected for the yield-related traits analyses were under varying in-field illumination conditions and weather conditions, containing a range of background objects during the experiments.

3.1.4.3 Wheat growth dataset for training, validation, and testing

Because images were collected from three consecutive years that cover four key growth stages (Figure 3.4), we decided to use the 2015 sequence (Figure 3.6) to train the models, because of the constant clarity and contrast of the image series. Then, we use the 2016 sequence (Figure 3.7) to validate our learning model and the final year, i.e. the 2017 sequence (Figure 3.8), to test the model. This training strategy gives us a reasonably robust validation of the performance of our model as the unseen sequence in 2017 can be utilised to test the generalisation of the model. Figure 3.5 illustrates the distribution of selected images in each growth stage in each growing season (30 images per year, 90 in total). Amongst these growth stages, the flowering stage has the highest number (37 out of 90), followed by heading (22 images), grain filling stages (19 images), and booting (12 images). The reason for this arrangement is that the flowering stage represents the phase when spikes are fully emerging, whereas wheat spikes are normally partially hidden at booting and heading stages (i.e. GS41-598). It is worth noting that the 2015 sequence does not

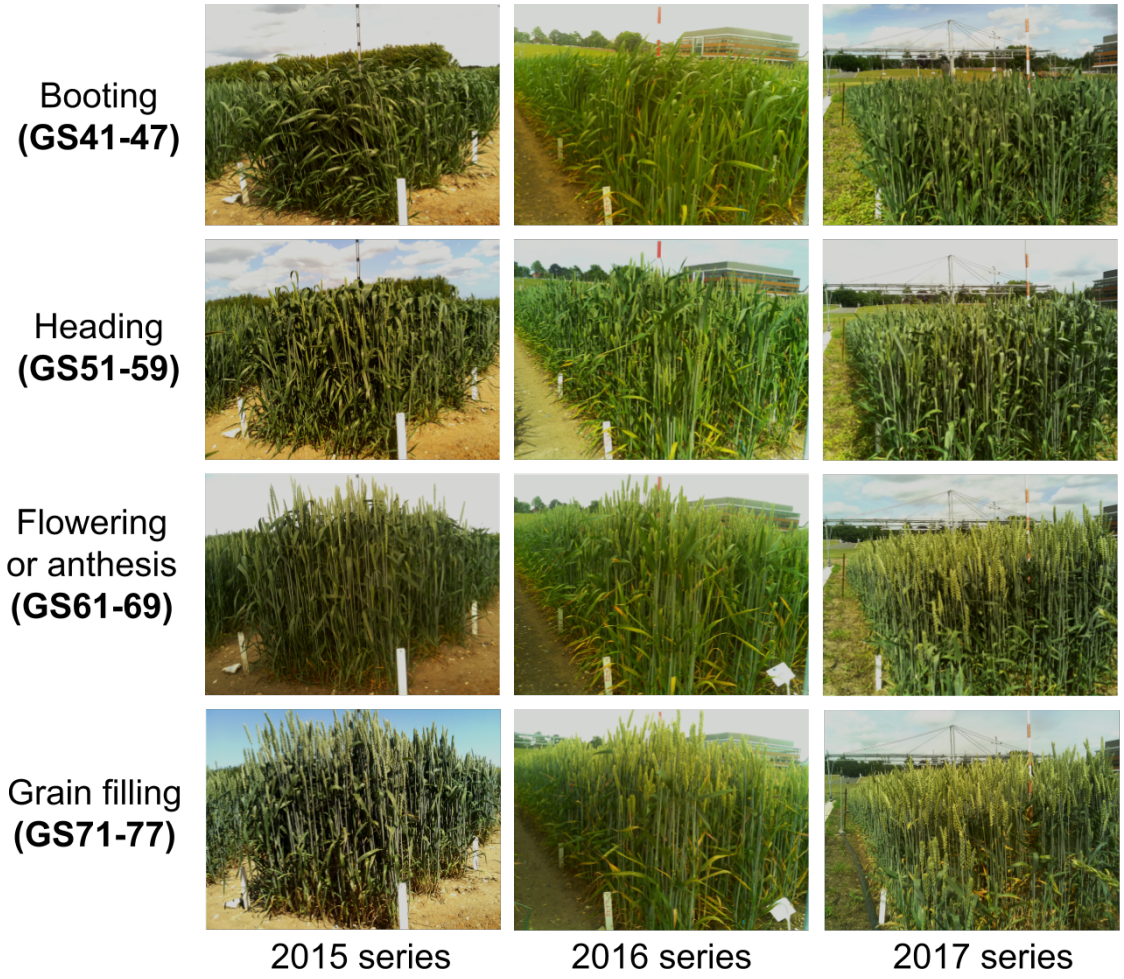

Figure 3.4: Wheat growth image series in the field collected by CropQuant workstations, from 2015 growing season to 2017 growing season, covering booting to grain filling growth stages

contain many booting images due to the short-term nature of wheat booting, which normally finishes within 1-2 days. Hence, it is an interesting test case for us to train a deep learning model that can segment spike regions collected in multiple years during the process of ear emergence (e.g. spikes have partially emerged) under challenging in-field lighting conditions. Figures 3.6, 3.7 and 3.8 show examples of 2015, 2016 and 2017 sequences images with the corresponding labels, the spike regions masks and spikelet density maps.

3.1.5 Wheat growth dataset Labelling

Using the UEA Computer Vision - Image Labelling Tool [41], we have manually labelled images with polygonal labels for the segmentation task and dot annotation for density estimation task. The tool runs as a web-based application and is written in Python 2.7. All images that need to be labelled are stored in a certain folder. Then, the tool detects and load these images. The tool provides a number of options to label the image. We selected the polygonal option to trace the spike shape. Once the tracing is done, we then select the appropriate label and then the corresponding coordinates of polygons are stored in matched encoding file for each image. For the density estimation task, we adjusted the tool to make the dot annotation option - which are represented by a circle- by reducing the size of the circle. The centre coordinates of each circle, representing each spikelet, are stored in the corresponding encoding file for each image.

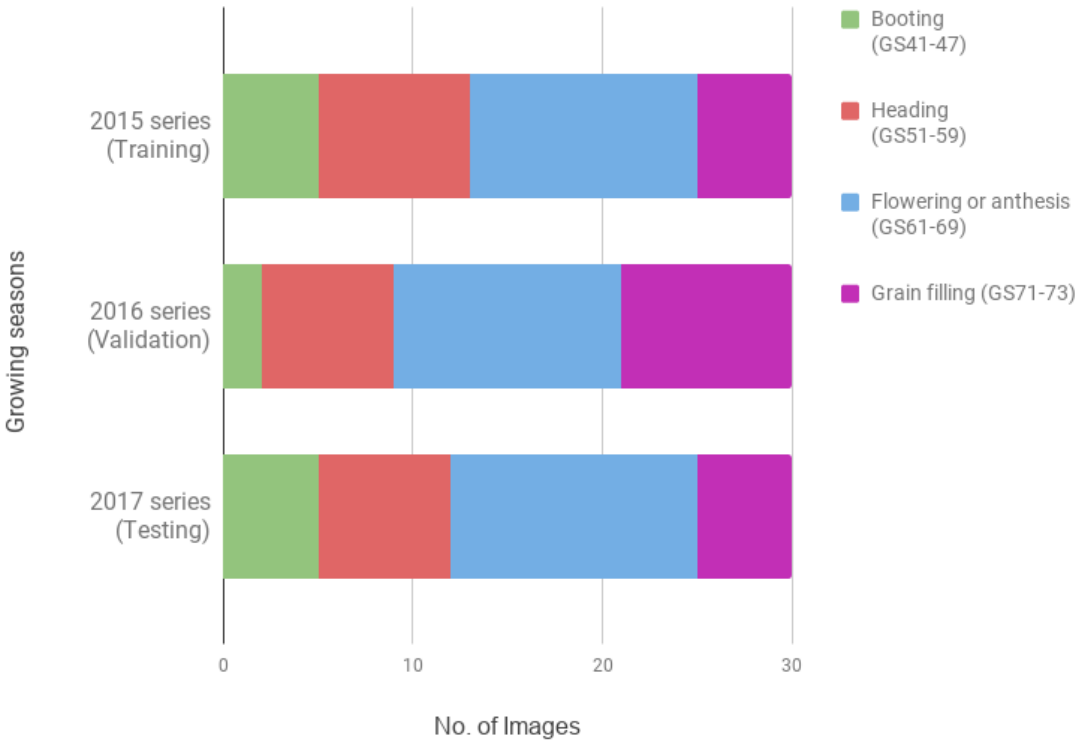

Figure 3.5: The distribution of selected images in each growth stage between 2015 and 2017 growing season, which are used for training, validation and testing when establishing the deep-learning architecture

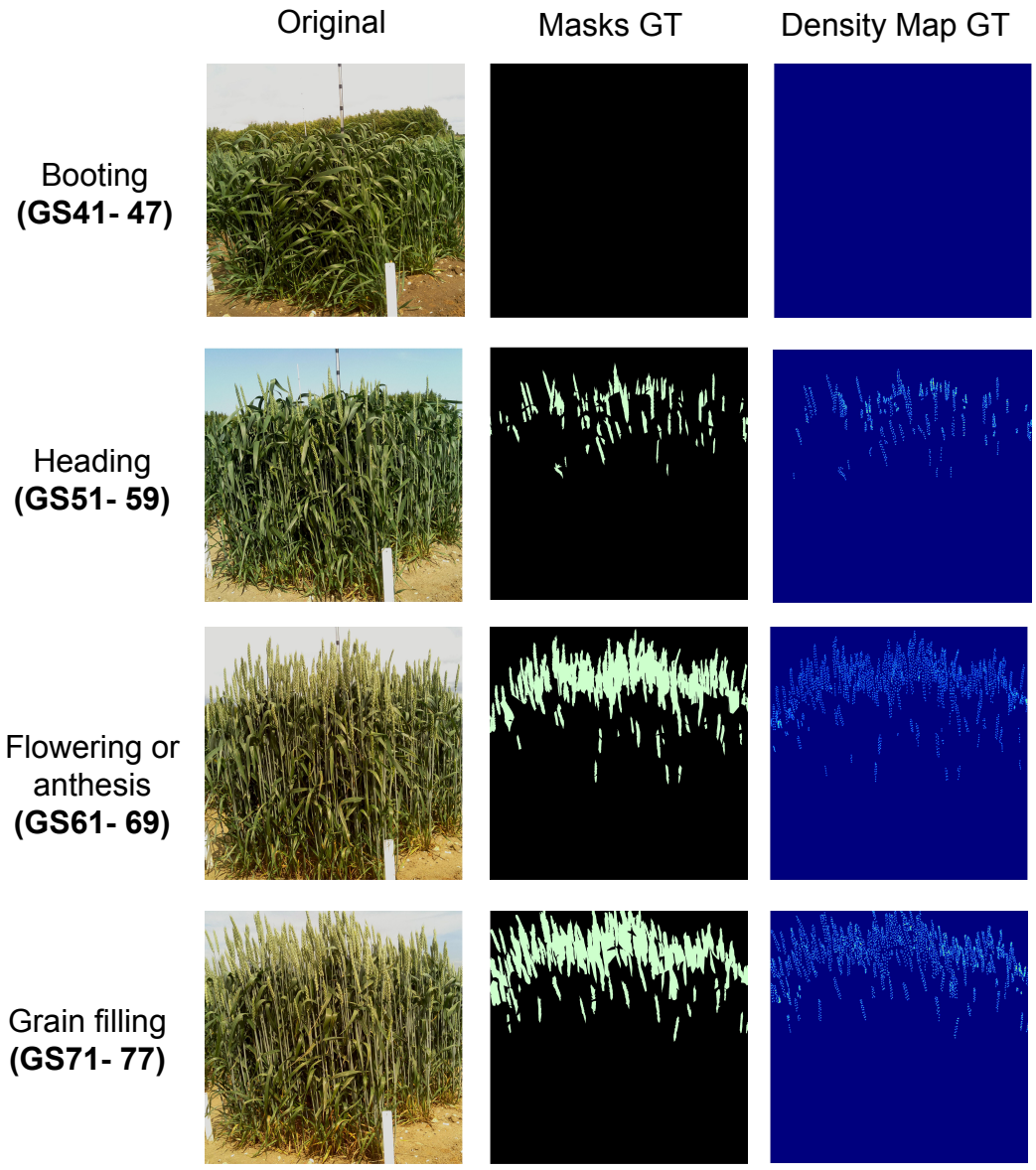

Figure 3.6: Wheat growth image series in the field collected by CropQuant workstations of the 2015 growing season, covering booting to grain filling growth stages and showing the corresponding spike regions segmentation masks and density estimation maps

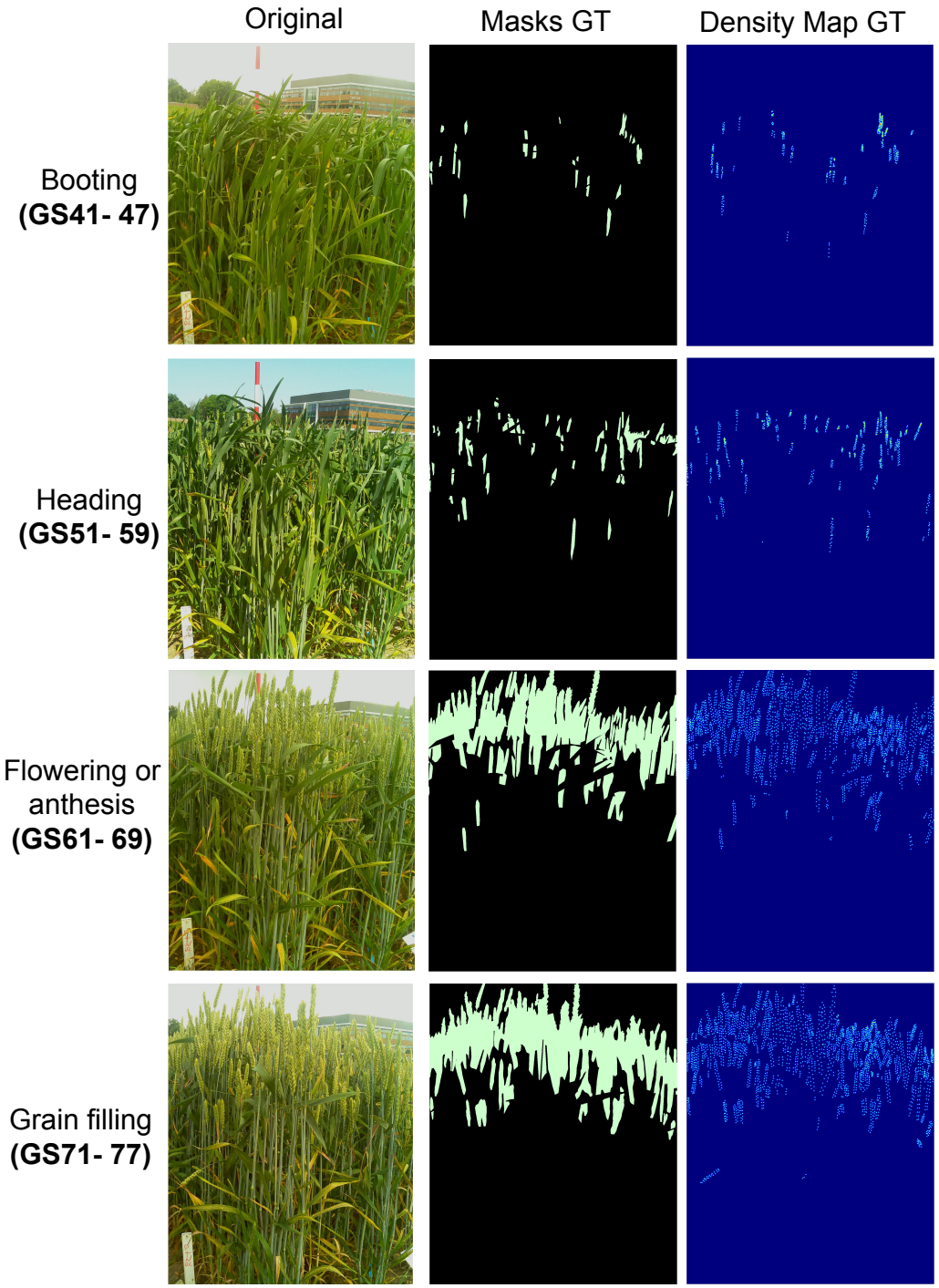

Figure 3.7: Wheat growth image series in the field collected by CropQuant workstations of the 2016 growing season, covering booting to grain filling growth stages and showing the corresponding spike regions segmentation masks and density estimation maps

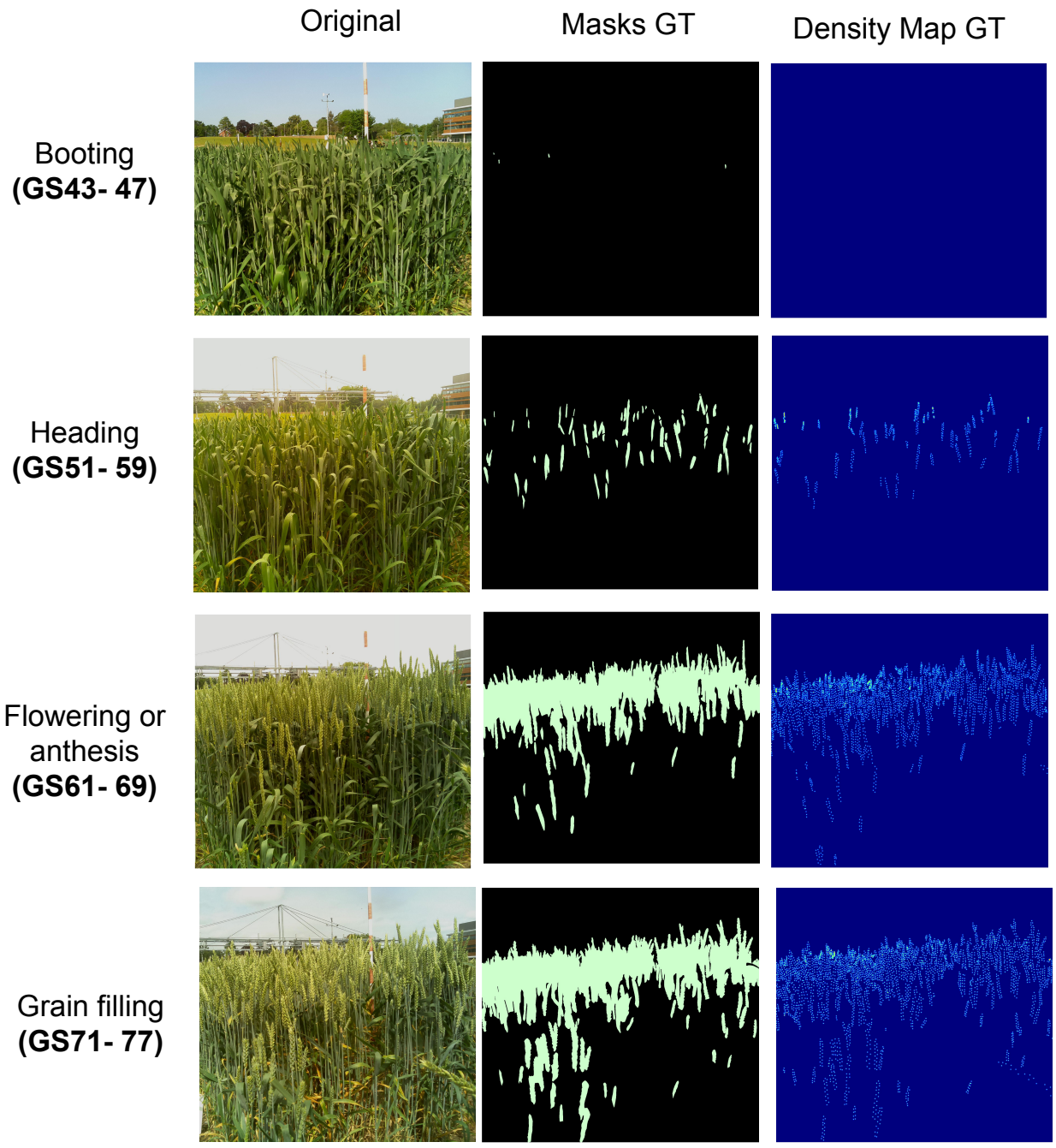

Figure 3.8: Wheat growth image series in the field collected by CropQuant workstations of the 2017 growing season, covering booting to grain filling growth stages and showing the corresponding spike regions segmentation masks and density estimation maps

3.1.5.1 The Annotated Crop Image Dataset (ACID)

The Annotated Crop Image Dataset (ACID)[90], which is publicly available has 520 images of wheat plants captured from 21 pots in a glasshouse with a resolution of 1956×1530. The imaging is done by 12 MP cameras and all images have a black background. The images show different spike arrangements and leaves and were obtained in consistent lighting. Also, the images were dot annotated by placing a dot in the centre of each spikelet. The total number of spikelets in all images is 48,000 and the average spikelet number per scene is 92.3 with a standard deviation of 28.52.

Figure 3.9 shows examples of both CropQuant and ACID images which exemplify their similarities and differences. For example all ACID images have a black background and consistent lighting whereas our images can contain a number of objects in the background as well as parts of the sky, variations in lighting, wind conditions, etc. Nevertheless, their dataset of dot annotated wheat images, which is publicly available, may be of use for our own models as the images represent the same physical entity as we are studying and may be used within Transfer Learning.

(a) (b)

Figure 3.9: Example of images from (a) ACID dataset and (b) CropQuant dataset

3.1.5.2 The protocol for training and validation

We randomly sampled sub-images from the original images for training and testing. Figure 3.3 explains a high-level workflow that we followed consisting of two steps:

1. Detecting Spike Regions using semantic segmentation where first we apply random sampling to extract patches from the original scene and corresponding mask. Then, we train SpikeSEG employing Fully Convolutional Net to learn non-linear transformation of spike regions segmentation. In testing phase: A Window, matching the sampled patch size, slides over the test image and SpikeSEG is applied to output a prediction map of spike segmentation.

2. Estimating Spikelet Count using density estimation where first, as in the previous step, we apply random sampling to extract patches from the original or Optimal scene and corresponding dot annotation. Second, we apply Gaussian smoothing filter ($\sigma = 2.5$) to each dot to generate the density of each spikelet which can be summed to one. Then, we train SpikeCount to perform spikelet density regression for each pixel. Finally, to produce the spikelet count from a predicted density map, we simply perform integral operation(summation) over the whole map.

3.1.6 Employing Transfer Learning

Moving to the fifth component of our proposed methodology which is utilising a Transfer Learning technique to aid solving the problem of spikelets counting. Transfer Learning can provide immense advantage in improving and enhancing target task learning of any model. Traditionally, machine learning methods tend to learn the required task from scratch relying only on the training data [83]. However, when the training data is limited in quantity and quality, Transfer Learning can be employed to transfer the knowledge form previously learned tasks to be used to

aid learning the new target task [83]. We intend to use Transfer Learning in three main ways:

3.1.6.1 Transferring knowledge of ImageNet

In this context we are proposing to initialise SpikeSEG with ImageNet weights or parameters [103] and train it to segment the spike regions. ImageNet parameters are "off-the-shelf" pre-trained parameters obtained from millions of examples in thousands of object categories such as the ImageNet database [103] which will provide a general library of features that can be used for the first layers of any CNN model that could capture the low-level features of objects.

3.1.6.2 Transferring knowledge of ACID dataset

In the Subsection 3.1.4 we presented the fourth component of the proposed methodology which is the datasets used to achieve the main objective of this work. We propose to transfer the knowledge of Annotated Crop Image Dataset (ACID) [89] to be utilised when counting spikelets because of the similarity of the domain between the ACID and CropQuant dataset. Also, because the ACID dataset had spikelets dot annotated which provide immense advantage in favour of our main research objective. We will first train SpikeCount solely on the ACID dataset for learning the task of spikelets density estimation. Then we intend to investigate the effect of transferring the knowledge from the ACID dataset in two ways: (1) we will only continue training SpikeCount on CQ_2015, (2) we will freeze the encoder blocks gradually to investigate which part of the ACID features have positive impact on generating the best outcome.

3.1.6.3 Multi-task Learning (MTL)

Multi-task Learning (MTL) is a branch of machine learning, where several relevant targets are trained simultaneously to solve complex goals [15] where relying on

target modularity and solving each one independently can be ineffective for some problems. This approach can result in more accurate models for each learning task by sharing useful information learnt from relevant tasks [147]. Additionally, MTL is considered a branch of inductive Transfer Learning where the purpose is to find a low-dimensional feature representation which is shared among different targets [83]. As per the discussed workflow earlier, we propose two steps: (1)

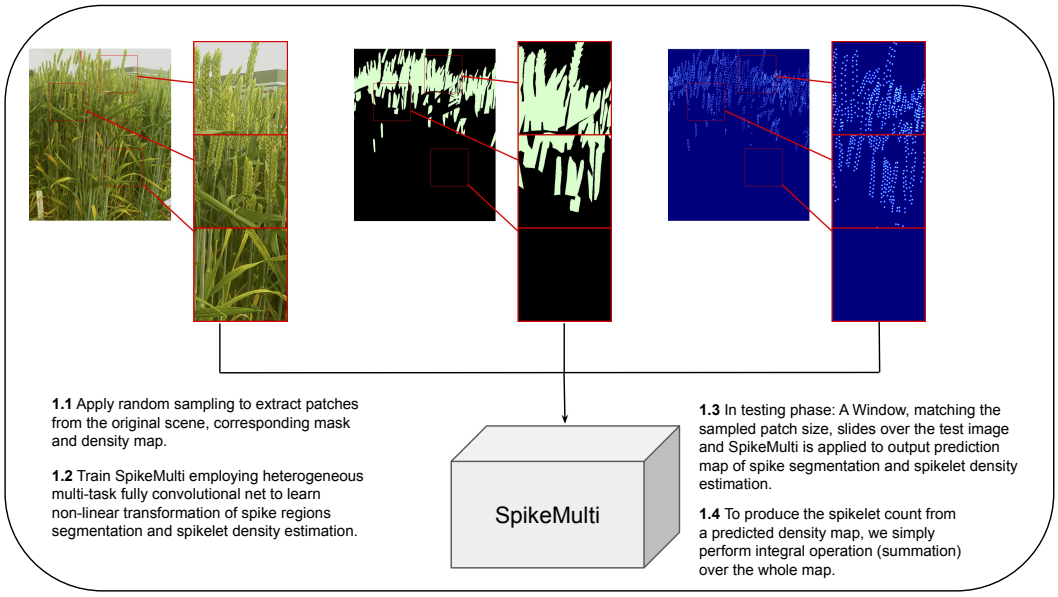

Figure 3.10: Illustration of multi-task learning developing SpikeMulti, a heterogeneous multi-task fully convolutional net to simultaneously train and test spike regions segmentation and spikelet density estimation. It also details the training, validation and testing strategy for developing SpikeMulti model

apply segmentation to isolate the Spike regions (SpikeSEG). (2) Apply spikelets density estimation on the product of the previous step to extract the number of spikelets(SpikeCount). This is considered the conventional workflow that we will implement. We will investigate its efficacy. We will also use MTL (SpikeMulti), which is illustrated in Figure 3.10, to simultaneously allow FCN to be trained to tackle both segmentation and density estimation and we will investigate the efficacy of this approach compared to the conventional way.

3.2 Testing and Evaluation

3.2.1 Testing

Similar to standard CNN approaches, a form of sliding window (tiling), as illustrated in Figure 3.3, is used to validate performance on the 2016 and then test on the 2017 sequence. We scan the target wheat scene sequentially with a window of a size equal to the subsampled patch size selected when training the model with a fixed stride (step size) of s equal to the window size. For each scanned region, the model is applied to predict for either spike segmentation or spikelet density estimation (or both in the case of the MTL). Then, the predicted sub-map will be copied in its appropriate location to create a prediction map. The window size corresponds to the sub-image size that is chosen by the experimental setting.

The result of the workflow is a prediction map varies according to the learning task. For example, the prediction output map of spike segmentation would be with a size of $w \times h \times cl$, where w and h correspond to the original image's width and height and cl is the number of classes, two in this case while the prediction map of spikelets density estimation would be with a size of $w \times h$, where w and h correspond to the original image's width and height.

3.2.2 Evaluation

3.2.2.1 Spike Segmentation Evaluation

To verify the result of the segmentation, we report the following metrics that are commonly used in semantic segmentation work [71, 62, 8]:

- Global Accuracy (GA) measures the total number of pixels that were predicted correctly over all classes divided by total number of pixels in the image.

The GA can be calculated using:

$$GA = \frac{\sum_i t_{pi}}{n_p} \tag{3.1}$$

where $\sum_i t_{pi}$ is the number of pixels that are predicted correctly for each class i and n_p is the total number of pixels in a given image.

- Mean class Accuracy (MA) is the mean of spike and non-spike region accuracy. The accuracy for each class can be calculated using:

$$ClassAccuracy = \frac{\sum_i n_{ii}}{t_i} \tag{3.2}$$

where $\sum_i n_{ii}$ is number of pixels that are predicted correctly to be of class i and t_i is the number of pixels of a certain class i.

- Mean Intersection over Union (MIoU) which is the mean of IoU of each class. MIoU is considered the harshest metric among all because of its sensitivity toward methods with a high false positive f_p rate or false negative f_n rate or both:

$$IoU = \frac{t_p}{t_p + f_p + f_n} \tag{3.3}$$

where, f_p, t_p and f_n denote respectively false positive, true positive, and false negative predictions. This metric was also used in the VOC PASCAL challenge [34]. In our case, it penalises methods that are more inclined towards predicting a spike-region pixel as background or vice versa. We reported the spike region and background measures separately for two reasons: (1) it is important to observe the model performance to recognise the spike region not the background; (2) it is clear that the ratio of background pixels to the spike region pixels is high, especially in early growth stages (i.e. booting and heading) where fewer or no spikelets can be found at the image level, indicating that some images can exhibit imbalanced class distribution. Consequently, it is important to observe evaluation measurements for both classes in the context of such class imbalances.

3.2.2.2 Spikelets Counting Evaluation

From the mathematical modelling of spikelets density estimation explained in subsection 3.1.1, we concluded that the total number of spikelets in a certain image I_i is the sum of all pixels densities in density map: $|P_i| = SPC_i = \sum_{p \in I_i} D^p$. Thus, for simplicity we will refer to the total number of spikelets inferred from the ground truth density map as SPC_i^{GT} and the total number of spikelets inferred from prediction density map as SPC_i^{PR}. Object counting methods use two evaluation metrics to measure the model performance when applied on testing images: Mean Absolute Error (MAE) (Equation 3.4) and Mean Square Error (MSE) (Equation 3.5).

$$MAE = \frac{1}{N} \sum_{i}^{N} \left| (SPC_i^{GT} - SPC_i^{PR}) \right| \tag{3.4}$$

$$MSE = \sqrt{\frac{1}{N} \sum_{i}^{N} (SPC_i^{GT} - SPC_i^{PR}))^2} \tag{3.5}$$

The MAE metric only measures the difference between predicted and GT spikelet number for each scene not taking into account the relationship between the difference and the GT. Therefore, we use the symmetric mean absolute percentage error (SMAPE) to estimate the percentage error (difference) of predicted spikelets number for a certain image and how that relates to the GT. In addition, using percentage error metrics make the results more readable and clear.

$$SMAPE = \frac{1}{N} \sum_{i}^{N} \frac{\left| (SPC_i^{GT} - SPC_i^{PR}) \right|}{\left| SPC_i^{GT} \right| + \left| SPC_i^{PR} \right|} \tag{3.6}$$

3.3 Summary

In this Chapter, we provided a formal definition of the problem of extracting key-yield traits such as spikelet numbers from wheat crop images captured images uncontrolled environments. First, we proposed to model the problem as crowd

counting and since the majority of the work that has been done in this area employs density estimation we follow that lead. This is because if we look closer to any wheat scene, we can see the similarity between these two problems (crowd counting, spikelets counting). Therefore, we adapted density estimation to count spikelets. Then, we introduce the workflow that consists of two steps: apply spike segmentation and then apply spikelets density estimation to estimate their numbers. Both these steps require fine grained output that matches the input dimension therefore for simplicity and because we want to investigate the MTL approach, we propose to use a fully convolutional network (FCN) and explain its advantages, particularly for spikelets density estimation. Then, we also describe MTL and its potential to provide reasonable solutions compared to the two step conventional workflow. We introduce the datasets we constructed to measure the efficacy of the different approach. We have relied on images that are collected from three consecutive years (2015, 2016, and 2017) that cover four key growth stages (Booting, Heading, Flowering and Grain filling). We also describe the ACID dataset and its importance for this work. We describe what is Transfer Learning and how we propose to use it to potentially improve the results of spike segmentation, spikelet density estimation and MTL. Finally, we present the evaluation metrics we use to measure the performance of the workflow proposed. The next Chapter (Chapter 4) will explain the experimentation setup of the first step in the workflow, spike segmentation, and discuss the results of different experiments.

Chapter 4

Spike Regions Segmentation

In Chapter 3, we outlined the overall proposed methodology to extract spikelet numbers from infield wheat crop images that were captured in a uncontrolled environment. As we established in Chapter 2, the majority of plant phenotyping solutions have been built on comparatively high-quality images and in more controlled environments. On the other hand, captured images of infield crop in complicated field conditions that are less controlled and are more challenging; hence, require more robust systems to extract accurate yield-related traits. For this reason, the first step in our proposed approach is to explore the idea of isolating ROI (i.e. spike regions) from noisy background by applying semantic segmentation so that sound phenotypic analysis could be carried out.

We propose to apply semantic segmentation using (SpikeSEG): a Fully Convolutional Network [71] to segment spike regions from wheat growth images based on annotated image data collected by CropQuant (CQ) in-field phenotyping workstations [152]. We employ the Transfer Learning approach by loading ImageNet[103, 62] parameters to improve the performance of the learning model, through which suitable features can be identified. In addition, we investigated effects of two input image sizes when training SpikeSEG on wheat images, as well as the model's performance at each key growth stage. To our knowledge, the SpikeSEG approach has not been applied to classify spike regions in field conditions. The result of our Spike-

SEG segmentation based on a three-year wheat growth image series is highly correlated with ground truth data that was manually labelled. Furthermore, through the evaluation of outputs of each max pooling layer in the learning architecture, novel vision-based features can be derived, and this is potentially important to assist crop scientists to visually debug and select features that are relevant to the feature selection procedure. We believe that the deep-learning approach presented in this Chapter could have important impacts on the current methodological development for measuring wheat spikes (i.e. spike regions on wheat growth images), in particular under real field conditions. Also, the phenotypic analysis workflow concluded during the exploration is likely to form an effective foundation to enable automated phenotypic analysis of key yield-related traits such as spike regions, key growth stages, and spikelets per unit area in the near future.

Section 4.1 summarise the datasets we used, the architecture of SpikeSEG, model optimisation, experimental setup, and training procedure details. Section 4.2 describes the performance results of testing SpikeSEG, and the their interpretation for each experimental setup. Also, we discuss results of testing SpikeSEG for each growth stage. At the end of the section, we take a closer look to the feature maps of the model by visualising its intermediate activation units and their interpretation. Finally, Section 4.3 presents our conclusions from these experiments.

4.1 Methods

In this section, we are going to explain SpikeSEG architecture, the specification of model training including the cost function, the training hyperparameters, and the protocol of splitting the sequences into training, validation and testing sets. In addition, we are going to explain in detail each experimental setup and its aim.

4.1.1 SpikeSEG architecture

The learning architecture of the SpikeSEG model established for segmenting spike regions is presented in Figure 4.1, which consists of four components:

- Very deep convolutional network: The first component of SpikeSEG is the so-called very deep convolutional network (VGG 16-layer net, VGG16 [115]). The segmentation-equipped VGG net (FCN-VGG16 or VGG16) has outperformed other classifiers such AlexNet51 and GoogLeNet [51] when it was selected as the base for FCN. It is a CNN classifier that achieved the first and second places in the ImageNet localisation and classification competition in 2014. Therefore, we have selected VGG16 as the base classifier for the task of spike segmentation. It has 12 convolutional layers arranged in five increasing convolutional depth blocks (Figure 4.1: (1) the first block, conv1, consists of two convolutional layers with a depth (number of filters) of 64; (2) the second block, conv2, consists of two convolutional layers with a depth of 128; (3) the third block, conv3, consists of three convolutional layers with a depth of 256; and, (4) the fourth and fifth blocks, conv4 and conv5 respectively, consist of three convolutional layers with a depth of 512. After each convolutional layer, there is a rectification non-linearity layer (ReLU) [26]. The filter size selected for all convolutional layers is 3×3 with a stride of 1. The reason for choosing such a small receptive field is that a non-linearity layer can be followed directly to make the model more discriminative [115]. After each block, a max-pool layer is added with a pooling size of 2×2 with a stride of 2. There are three fully connected layers at the end of the classifier. The first two fully connected layers, FC6 and FC7, have a depth (units) of 4,096. The depth of the last connected layer is 1000, which corresponds to the number of classes in the ImageNet competition [103]. The sixteenth (last) layer is the softmax prediction layer, which comes after the last connected layer. It is worth noting that the last connected layer is removed in our architecture

as our task requires prediction for two and not 1000 classes.

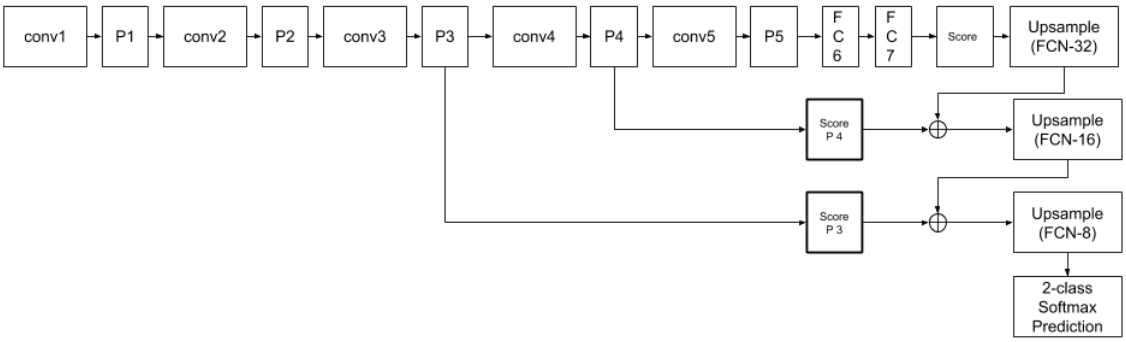

Figure 4.1: The SpikeSEG architecture established for segmenting wheat spike regions.

- Fully convolutional layers: The second component of FCN is transforming the first two fully connected layers FC6 and FC7 in VGG16 to convolutions. This setting will restore all the spike regions' spatial information that might be lost with the fully connected layers.

- Deconvolutional layers and feature fusion: Even though restoring the spikes' spatial details can help with the segmentation task that involves predicting dense output, the output from the fully convolutional layers is still coarse due to the repeat application of convolutions and subsampling (max-pool), which reduces the output size. In order to refine the coarse output and retain the original resolution of the input, the model fuses the learned features from three positions in VGG16 with the upsampling layers. Upsampling or deconvolutional layers reverse the effect of the repetitive application of subsampling and convolving by learning backward convolution. In order to apply the fusion operation, three prediction layers were added: (1) after the last fully convolution layer FC7, (2) after the fourth max-pool P4, and (3) after the third max-pool P3. The reason for predicting at different positions is to fuse lower level information obtained from the lower layers together with higher level information obtained from the higher layers, which can further refine the output. Next, the output of the first prediction layer is upsampled by applying the first deconvolutional layer. Then, the first upsampled output

(FCN-32) is fused with the second prediction layer (Score P4) by applying element-wise summation. It is worth noting that a cropping operation is applied to the upsampled output, so it matches the size of the second prediction output. Then, the output would be upsampled using the second deconvolutional layer (FCN-16) to be fused with the output of the last prediction layer (Score P3). Lastly, a final deconvolutional operation is applied to the output to be upsampled to the input size of the original image (FCN-8).

- Softmax layer: The last layer of SpikeSEG is a 2-class softmax[63] calculating the probability of each pixel for each class.

4.1.2 Cost function

According to any common semantic segmentation task [71], for each pixel x_{ij} in an image I with a size of $h \times w \times d$, a corresponding pixel label class t_j from a probability distribution 0, 1 is assigned. The predicted class of a certain pixel y_{ij} is the outcome of the last softmax layer, which generates a probability distribution such that $0 \leq y_{ij} \leq 1$. The learning task is to find a set of parameters (i.e. weights) θ that, for a particular loss function $\mathrm{l}(y_{ij}(x_{ij}\ ,\ \theta))$, will achieve the minimum distance of the probability distribution between the target class t_j and the predicted class y_j. The cost function used here is cross entropy, L, which calculates the negative log likelihood of the predicted class y_j:

$$L = -\sum_{j=1}^{m} t_j \log y_j \tag{4.1}$$

where m is the number of classes and in our case is 2, corresponding to spike region versus background.

4.1.3 Training hyperparameters

Hyperparameters need to be initialised before the training process starts. Then, the training algorithm learns new parameters as part of the learning process [63]. The SpikeSEG training hyperparameters values used in our study are listed in Table 4.1, including:

- Weight θ (parameters) /Bias initialisation: It is good practice when training any deep learning model from scratch to initialise the weights with random values and the bias with 0. There are many initialisation techniques that can be used but we found He et al. [51] achieved the best results when training from scratch. Their technique generates a mean centred normal distribution with standard deviation σ equal to $\sqrt{2/n_l}$ where n_l is the number of inputs in a certain layer l.

- Dropout rate probability: This parameter serves as a regulariser to reduce the model overfitting [118]. It determines how many units can be deactivated randomly for every training iteration in a certain layer. In our model, we added two dropout layers, with a value of 0.5 for p, which represents the probability of keeping the unit activation or not. This is recommended when using layers with large numbers of units [118] such as FC6 and FC7. Therefore, the dropout layers are added after every fully convolutional layer FC6 and FC7.

- Intermediate non-linearity unit: This is an essential component in any CNN that focuses on highlighting and emphasising the relevant features of the data and the task. As a default, we have selected Rectified Linear Unit (ReLu) for this parameter which is an element-wise thresholding operation that is applied on the output of the convolutional layer (resulting feature map) to suppress negative values: F(x)=max(0,x) where x is an element in the feature map.

- Epochs: This refers to the number of training iteration, which was set to 125-150 according to the early stopping method.

- Optimisation algorithm: We have tried many optimising algorithms and mini-batch stochastic gradient descent (SGD) was more suitable for the pixel classification problem we have. Hence, the weights are updated for every learning iteration using mini-batch (SGD) with momentum: $v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta), \theta = \theta - v_t$ [91]. The initial learning rate was chosen as 0.001 with a decay of 0.0016 for every epoch. The momentum Y is the default 0.9 and the selected mini batch is 20.

  We have selected the initial learning rate by experimenting with lowest learning rate (1×10^{-5}) and linearly increasing it until we reached the optima, i.e. when we reached the lowest validation loss in conjunction with selecting different values of learning decay.

Table 4.1: SpikeSEG Training Hyperparameters

Stage	Hyperparameter	Value
Initialisation	Weights	He et al. [51] (scratch) ImageNet (transfer)
	Bias	0
Dropout	Rate p	0.5
Intermediate Non Linearity Unit	Relu	
Epochs	125 – 150	
Optimisation (SGD)	Learning Rate	0.001
	Momentum	0.9
	Decay	0.0016
	Mini Batch	20

4.1.4 Training and validation of the architecture

We have selected the 2015 dataset for training SpikeSEG and the 2016 dataset as the validation set to observe if there is overfitting of the model. However, because of the high resolution of the images (2592×1944) , we randomly sampled 450 sub-images with a size of 512×512, and 8999 sub-images with a size of 128×128 with their corresponding manual labels. This is to evaluate which resolution will lead to the best results. This approach is to reduce computational complexity, as dealing with the original image size would be unmanageable in the training process. We have utilised an early stopping technique when training the model. Early stopping allows us to keep a record of the validation learning (e.g. cost and accuracy) for each learning epoch. It is a simple and inexpensive way to regularise the model and prevent overfitting as early as possible [63, 9]. We have selected the validation cost as the metric to observe for early stopping. The maximum epochs for observing the change in validation cost is 20 epochs. In other words, if the validation cost has not been decreased for 20 epochs, the model training will be stopped and the model weights resulting from the lowest validation cost are saved. We have found that the model for all our experimental trails converge after training for 125 to 150 epochs.

In addition to train SpikeSEG from scratch, we wanted to investigate whether the Transfer Learning approach [103] can produce improvements of the validation accuracy. One of the advantages of using deep segmentation architectures that are built on top of state-of-the-art classifiers is that we can apply Transfer Learning. Transfer Learning can be described as using "off-the-shelf" pre-trained parameters obtained from millions of examples in thousands of object categories such as the ImageNet database [103]. These parameters represent a general library of features that can be used for the first layers of any CNN model, since the first layers are only capturing the low-level features of objects (corners, borders, etc.) It is then possible to only fit the higher-level layers of the CNN that are more task and

data oriented. Therefore, we can initialise the CNN model with the pre-trained parameters and proceed with training the higher layers instead of initialising with random values and training from scratch. The application of Transfer Learning is extremely beneficial when there are limitations in the sample size and/or variation of example datasets as those are essential to train any sound deep architecture. Therefore, for our work, we have loaded the pre-trained weights from the ImageNet challenge to the VGG16 and then trained the model with the same hyper-parameter settings described previously.

4.2 Results

We evaluate the performance of SpikeSEG on both 2016 and 2017 datasets. The evaluation is conducted to test the segmentation performance of SpikeSEG considering multiple experimental set-ups. For example, the use of pre-trained parameters when training the model (Transfer Learning) is compared with training from scratch and the use of different sub-image sizes is also compared. Furthermore, we compared the performance on each growth stage separately as this might discover interesting interconnections between the monitored growth stages that have strong correlation to the grain production. We reported the spike region and background measures separately for two reasons: (1) it is important to observe the model performance to recognise the spike region not the background; (2) it is clear that the ratio of background pixels to the spike region pixels is high, especially in early growth stages (i.e. booting and heading) where fewer or no spikelets can be found at the image level, indicating that some images can exhibit imbalanced class distribution. Consequently, it is important to observe evaluation measurements for both classes in the context of such class imbalances. We also discuss the results of experimenting with different sub-image sizes.

4.2.1 Transfer Learning

The results in Tables 4.2 and 4.3 report on the experiments comparing the SpikeSEG model trained from scratch and with parameters learned from the reported ImageNet classification51 task on the segmentation of the 2016 and 2017 datasets respectively. Table 4.2 shows that MA and MIoU have been improved by 3.21% and 4.19% respectively when using the pre-trained parameters in the 2016 set. Particularly, the results of spike regions show an increase in both Spike Accuracy and spike IoU of 5.62% and 6.77%.

Table 4.2: Quantitative results of segmentation performance for the 2016 dataset when training SpikeSEG from scratch by initialising the weights using He et al. method [51] and by loading pre-trained ImageNet parameters showing different evaluation metrics where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

Initialisation	GA	MA	Spike Accuracy	MIoU	Spike IoU
He et al. [51]	91.84%	82.90%	70.22%	73.43%	55.98%
ImageNet Parameters	**93.34%**	**86.11%**	**75.85%**	**77.63%**	**62.75%**
change as %	1.51%	3.21%	5.62%	4.19%	6.77%

The results in Table 4.3 illustrate that MA and MIoU have improved by 6.33% and 6.83% in the 2017 set when using the pre-trained parameters. Notably, the results of the spike region show an increase in both Spike Accuracy and spike IoU of 12.57% and 11.66% respectively.

Table 4.3: Quantitative results of segmentation performance for the 2017 dataset when training SpikeSEG from scratch by initialising the weights using He et al. method [51] and by loading pre-trained ImageNet parameters showing different evaluation metrics where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

Initialisation	GA	MA	Spike Accuracy	MIoU	Spike IoU
He et al. [51]	90.46%	73.15%	48.15%	66.69%	43.69%
ImageNet Parameters	**92.47%**	**79.48%**	**60.72%**	**73.52%**	**55.35%**
change as %	2.01%	6.33%	12.57%	6.83%	11.66%

From the above results, it is clear that Transfer Learning has a positive effect on improving performance for both validation and test datasets. To further verify this finding, we present Precision-Recall curves in Figure 4.2 for each growth stage for the test and validation datasets. The left most sub-figures show two graphs that represent the Precision-Recall curves of the models trained from scratch, whereas the right most graphs represent the curves after loading ImageNet parameters. The

top two graphs refer to the 2016 validation dataset, whereas the bottom graphs present results for the 2017 dataset. Although there are relatively subtle improvement in early growth stages due to the limited sample size (green and red series in graph), it is noticeable that the Transfer Learning produces a "lift" effect on the Precision-Recall curves in both years. It is also evident that performance is particularly improved for later growth stages (from flowering inwards, blue and pink series in graph, when spikes were fully emerged). Given the positive effect of Transfer Learning, we used this approach in more detailed analyses on different sub-image sizes and growth stages.

Figure 4.3 shows the segmentation performance using MA and MIoU for the 2016 and 2017 image series, when training SpikeSEG by loading pre-trained ImageNet parameters. The Y-axis represents the values of MA/MIoU (in percentage) and X-axis represents the image ID arranged by its associated growth stage from 2016 to 2017, the smaller ID the earlier growth stage in the growing season (i.e. booting or heading). Figure 4.3 A indicates that MA and MIoU are relatively similar in all images, but there is a trend with growth stage as the earlier growth stages achieved lower evaluation metrics scores and the later growth stages achieved higher metrics scores. However, Figure 4.3 (b) does not show a similar trend in 2017, instead both metrics scores are fluctuating in values across the monitored growth stages. This may indicate that the images in the 2017 are more challenging, for example, more unexpected objects in the field, less image clarity, and changeable lighting conditions.

4.2.2 Different sub-image sizes

Tables 4.4 and 4.5 illustrate a comparison of two different sub-image resolutions, 128×128 and 512×512, for spike segmentation on the 2016 and 2017 datasets. In both cases, for almost all measures, the larger sub-image sizes produce better performance. For the 2016 set, the MIoU and Spike IoU have increased by 4.18% and 7.63% respectively using the 512x512 sub-image size, whereas the MA and

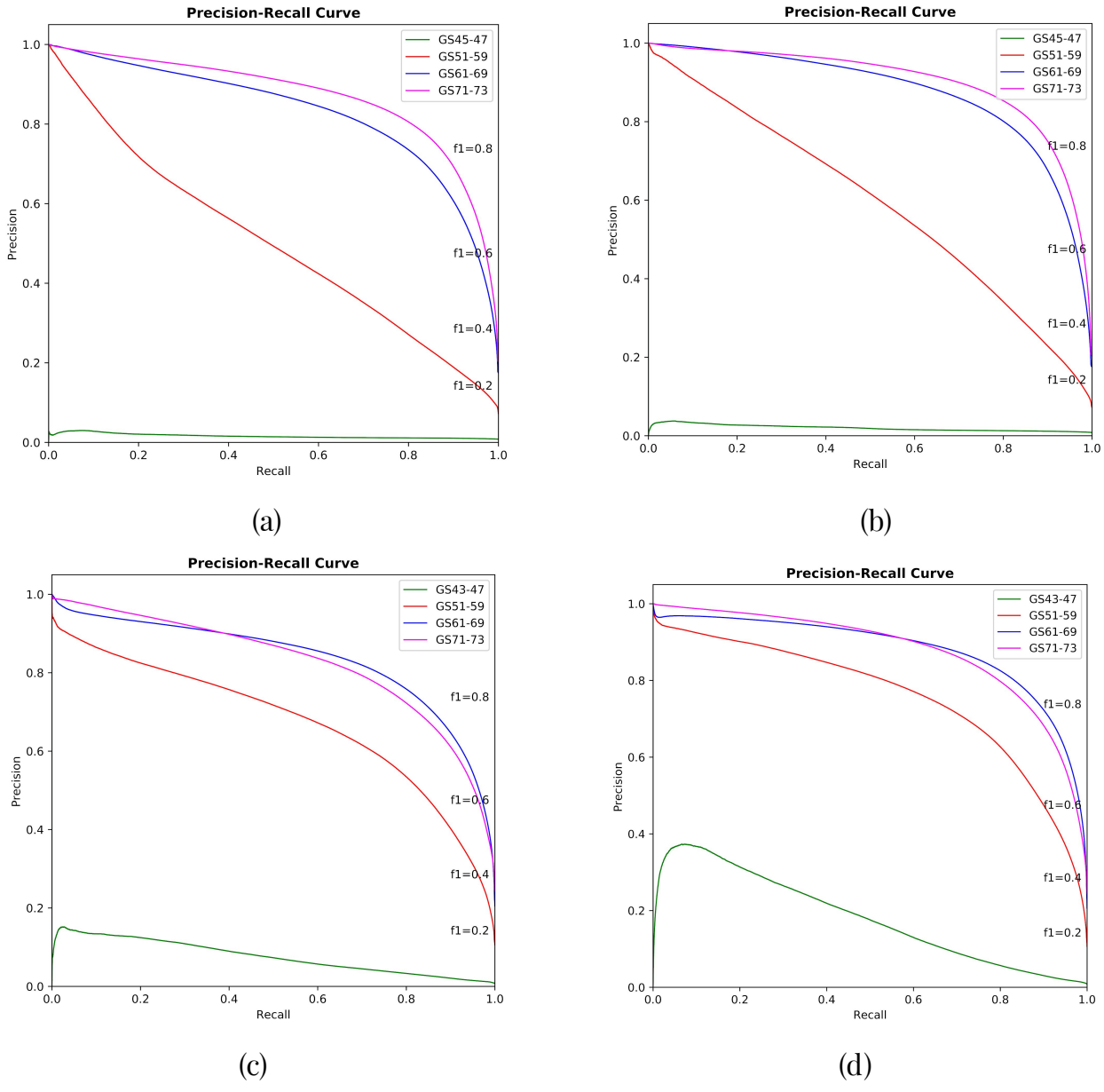

Figure 4.2: Precision-Recall curves showing the segmentation performance. Training from scratch 2016 series (a) and loading pre-trained ImageNet parameters series 2016 series (b) to report the segmentation performance at different monitored growth stages. Training from scratch series (c) and loading pre-trained ImageNet parameters (d) using series in 2017.

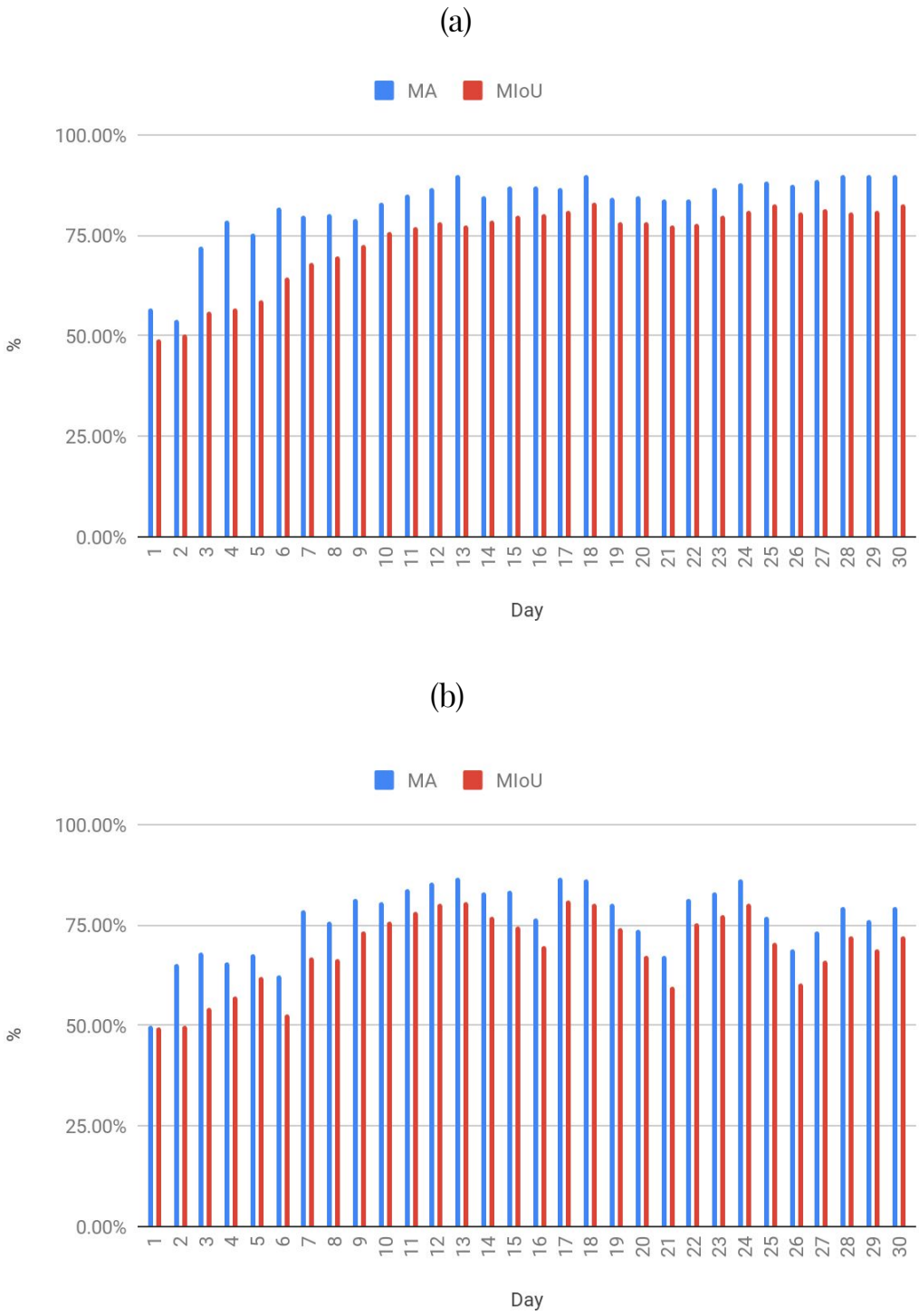

Figure 4.3: Quantitative results (MA and MIoU) to assess segmentation performance from day 1 to day 30. (a) The 2016 images trained by SpikeSEG using pre-trained ImageNet parameters. (b) The 2017 image dataset trained by SpikeSEG through loading pre-trained ImageNet parameters.

Spike Accuracy have improved by 6.14% and 13.68%. For the 2017 set, the MIoU and Spike IoU have increased by 10.37% and 18.25% using the 512x512 sub-image size, and the MA and Spike Accuracy have improved by 10.47% and 21.81%. As a result, we can see that selecting a larger sub-image size is likely to lead to better results based on the selected segmentation metrics.

Table 4.4: Quantitative Results of Segmentation performance for the 2016 dataset when training SpikeSEG with two different sub-image size S (i.e. 128x128 and 512x512) showing different evaluation scores where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

S	GA	MA	Spike Accuracy	MIoU	Spike IoU
128×128	92.52%	79.98%	62.17%	73.44%	55.13%
512×512	**93.34**%	**86.11**%	**75.85**%	**77.63**%	**62.75**%
change as %	0.83%	6.14%	13.68%	4.18%	7.63%

Table 4.5: Quantitative Results of Segmentation performance for the 2017 dataset when training SpikeSEG with two different sub-image size S (i.e. 128x128 and 512x512) showing different evaluation metrics scores where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

S	GA	MA	Spike Accuracy	MIoU	Spike IoU
128x128	89.86%	69.01%	38.91%	63.15%	37.09%
512x512	**92.47**%	**79.48**%	**60.72**%	**73.52**%	**55.35**%
change as %	2.61%	10.47%	21.81%	10.37%	18.25%

4.2.3 Phenotypic analysis of yield and growth traits

In Table 4.6, we report the spike segmentation result according to the growth stages to further investigate SpikeSEG's performance for each growth stage in 2016. Note that the 2016 dataset does not contain early or middle booting and hence we could only test late booting. Notably, the model performed very well in both flowering and grain filling stages. For example, in the grain filling stage, the MA and MIoU are 88.27% and 81.09%, respectively; whereas in the flowering stage, the MA is 86.24% and the MIoU is 78.92%. In the heading stage, the model has also achieved good results with the MA and MIoU equal to 79.21% and 65.91%. However, SpikeSEG has not led to good results in booting, where the MA is 54.69% and IoU is 49.75%. This is not surprising as not enough representative images for this stage were available in the training data.

Table 4.6: Quantitative results of segmentation performance of SpikeSEG for the 2016 dataset reported for each growth stage where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

Growth Stage	GA	MA	Spike Accuracy	MIoU	Spike IoU
Booting (GS45-47)	96.90%	54.69%	11.88%	49.75%	2.59%
Heading (GS51-59)	92.94%	79.21%	63.17%	65.91%	39.22%
Flowering (GS61-69)	93.12%	86.24%	75.68%	78.92%	65.77%
Grain filling (GS71-73)	93.17%	88.27%	80.05%	81.09%	70.33%

Table 4.7: Quantitative results of segmentation performance of SpikeSEG for the 2017 dataset reported for each growth stage where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

Growth Stage	GA	MA	Spike Accuracy	MIoU	Spike IoU
Booting (GS43-47)	98.84%	67.14%	35.01%	57.78%	16.72%
Heading (GS51-59)	94.02%	81.03%	64.61%	73.31%	53.03%
Flowering (GS61-69)	91.09%	80.87%	63.65%	74.48%	59.20%
Grain filling (GS71-73)	87.54%	75.62%	52.65%	68.07%	50.42%

In Table 4.7, we report the spike segmentation results based on the wheat growth stages in 2017. The table shows that the model performed well in the flowering stage with the MA equal to 80.87% and MIoU equal to 74.48%, which is likely achieved due to more image data present in this stage in 2015. The heading stage results and the grain filling stage are similar to the flowering stage. However, the model performed less well on the booting stage, corresponding to the lack of data for this stage in the training set. The results show that SpikeSEG performance increases with the development of spikes and it performs better if more representative training data can be included when developing the learning model.

It is worth noting that for both the 2016 and 2017 results, the GA values for the booting stage are higher compared to the other stages, which is not the case for any other evaluation metrics. This may be caused by the majority of the pixels being background in early growth stages, as those are predicted correctly by the GA metric, which focuses on predicting the sum of pixels regardless of the class. It does, however, reinforce the need for more than one single evaluation metric to assess the fitness of learning models as the GA value may not truly reflect the ability of the model during the segmentation. Figures 4.4 and 4.5 illustrate visualisation of the segmentation result for the 2016 and 2017 image series respectively.

4.2.4 Visualisation of SpikeSEG Intermediate Activations

In order to understand and interpret more about the features that SpikeSEG is utilising when testing wheat sub-images, we have visualised feature maps that are output by each layer in the SpikeSEG in the first five blocks (conv1- conv5) [142]. As illustrated in Figure 4.6, the sub-image chosen is from image ID 215 (see supplementary data), which scored the highest spike accuracy amongst all images. To simplify the presentation, we only show a number of feature maps that are output by three layers (i.e. Conv1 Block Maxpool, Conv3 Block Maxpool, and Conv5 Block Maxpool), where regions that are coloured from bright yellow to green indicate where SpikeSEG is activated, whereas the darker colour shows regions that are being ignored by the SpikeSEG.

Each square represents a feature that was activated by some part of the image. For example, we can observe that early layers of SpikeSEG (Conv1, max-pooling output) are activated by the spikelets or part of spikelets (outlined in blue square as an example). On the other hand, some features are activated by other objects in the background such as leaves,trees, clouds and sky (outlined in red square as an example). However, they show very low-level detail information, correlating with the fact that early layers in CNNs capture the lower level of features such as edge and corner-featured objects. The next feature maps (Conv3, max-pooling output) show that the SpikeSEG is more focused on the shape and texture-based features, which are considered higher level abstract features. The last feature maps (Conv5, max-pooling output) shows that the SpikeSEG is only preserving the general size- and texture-based features of spike regions as the low-level information has been lost due to repetitive application of pooling operations. In addition, image comparison with original images suggests that the SpikeSEG not only recognises spike regions, but also captures other background objects such as sky, soil, and leaves throughout these layers, which leads to segmentation results in Figures 4.4 and 4.5.

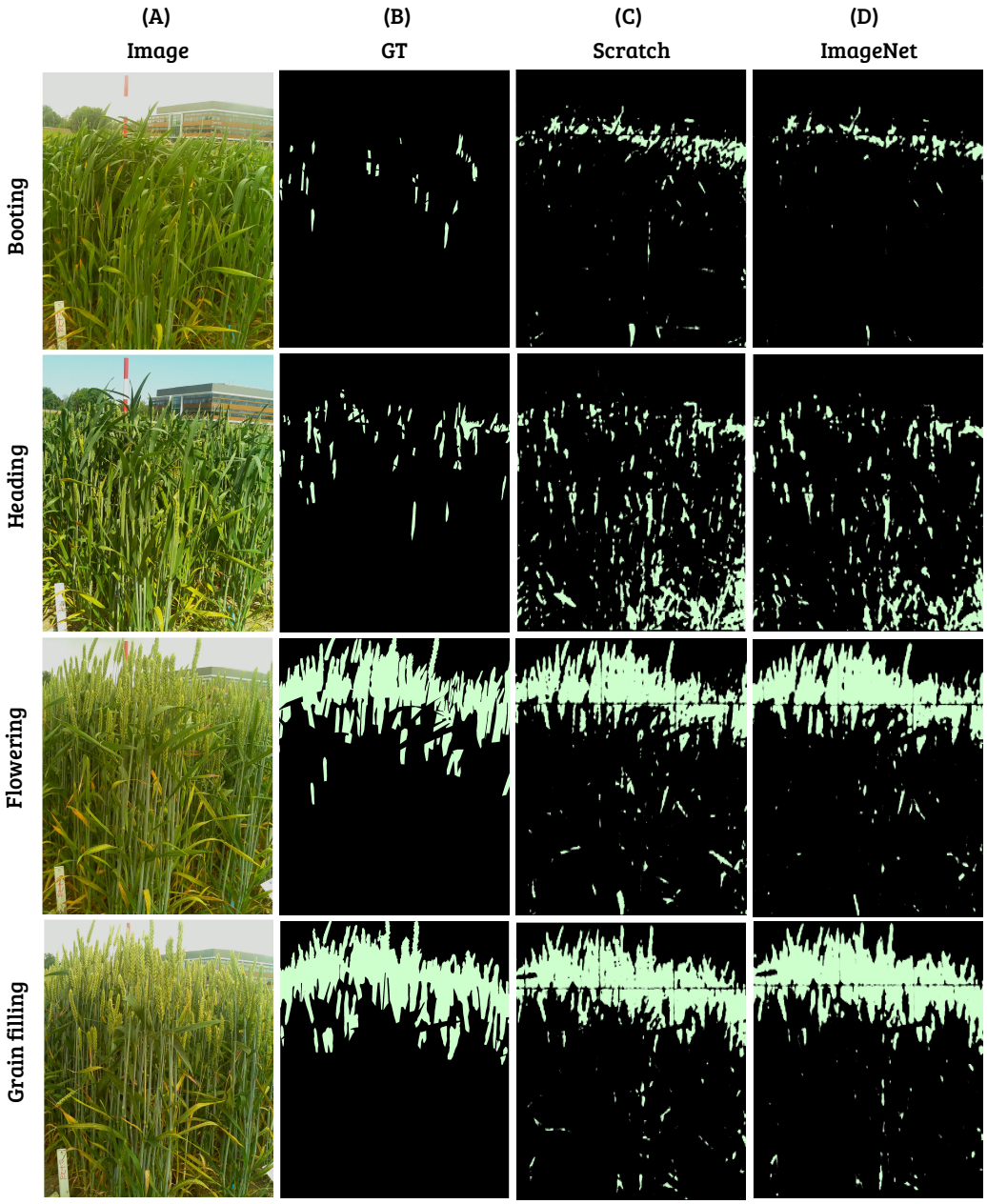

Figure 4.4: Visualisation of the segmentation result for the 2016 image series. (A) Original image, (B) Ground truth (GT), (C) The result of trained SpikeSEG from scratch, (D) The result of trained SpikeSEG by loading pretrained ImageNet parameters (from top to bottom, images were selected to represent different key growth stages)

4.3 Summary

In this Chapter, we have explored a method that combines deep learning and computer vision to discriminate wheat spike regions on wheat growth images through

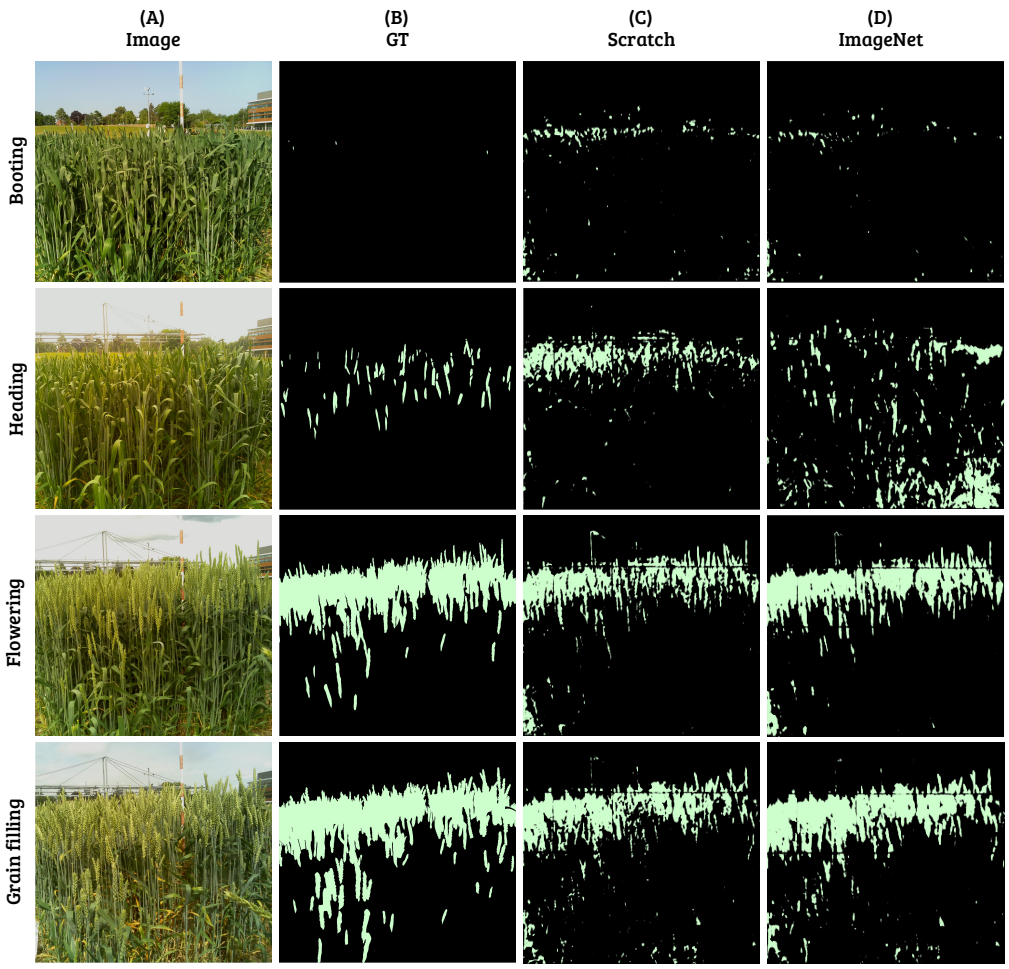

Figure 4.5: Visualisation of the segmentation result for the 2017 image series. (A) Original image, (B) Ground truth (GT), (C) The result of trained SpikeSEG from scratch, (D) The result of trained SpikeSEG by loading ImageNet parameters. (from top to bottom, images were selected to represent different key growth stages)

a pixel-based segmentation. This method was implemented using Python with a TensorFlow backend, which provides the framework for us to establish the SpikeSEG architecture. We can then move from the training phase to the final 2-class prediction at the image level. Our goal was to obtain a classifier that can analyse wheat spike regions using the standard deep learning approach, with little knowledge of wheat spike dimensional and spatial characteristics. We fulfilled this requirement by establishing a SpikeSEG model to segment spike regions in wheat growth image series acquired in three consecutive years, with varied weather conditions. The spike regions in all images have been annotated at pixel level by

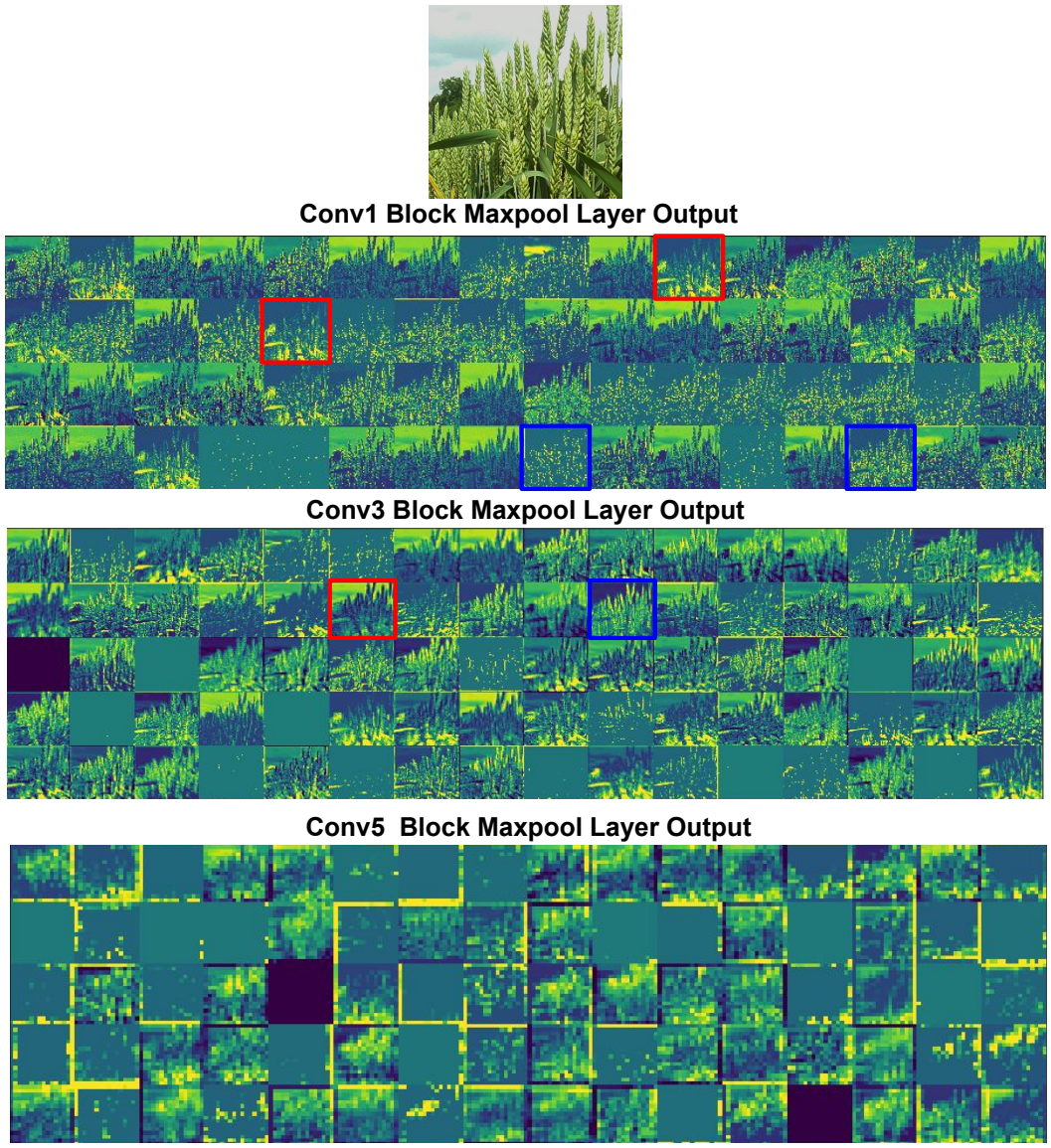

Figure 4.6: The selection of filters of three intermediate layers (Conv1, Conv3, and Conv5 block Maxpool outputs) showing activated features that could be used for visually assessing SpikeSEG on wheat sub-images. The squares outlined in blue are examples of features activated by spike/spikelets ROI where the ones outlined in red are examples of features activated by background objects.

specialists. The model performance was verified on both validation (the 2016 image set) and test (the 2017 image set) datasets. We have found that SpikeSEG was relatively successful at detecting the spike regions in both 2016 (MA: 86.11%) and 2017 (MA: 79.48%). In addition, SpikeSEG performed better when trained on larger sub-images sizes. We then applied Transfer Learning to improve the performance of our SpikeSEG model by loading parameters learned from ImageNet,

and this has led to a positive impact on the segmentation results. The limitations of our research can be summarised by three points: (1) the model had limited success when identifying spike regions in booting and heading; this may be caused by a lack of training data at the two stages; (2) the model encountered some unexpected background objects such as grass, and this has increased false positive rates; again, we believe that more training data or data augmentation could resolve this issue; (3) the model performed relatively poorly on the 2017 set due to challenging lighting and weather conditions. We might be able to overcome some of these image-based limitations by including more historic or artificial images in the training set as well as exploring other deep learning segmentation architectures such as DeepLapV3+[24] and also some traditional CV and ML segmentation methods. We will also trial other learning tasks in a MTL environment to improve the soundness of the solution.

Chapter 5

Counting Spikelets using Density Estimation

In Chapter 4, we have discussed the development of SpikeSEG model, a fully convolutional network, used to tackle the task of spike segmentation from infield wheat images. The aim of the spike segmentation task is to eliminate the background objects and minimise the challenging task of estimating the number of spikelets.

In this Chapter, we focus on one specific task that can help wheat management through monitoring systems: the task of counting spikelets in wheat images as a form of yield quantification for wheat crops.

Object counting from images is a difficult problem that emerges in many different scenarios, for example, monitoring crowds [92, 18, 143], performing wildlife census [6], counting blood cells in images [47] and others. Sometimes, the problem is framed in the context of counting objects in still images [5, 21, 6] whereas in other cases, the problem involves counting moving objects in videos [40].

Counting objects can be achieved by a number of approaches including counting by detection [134, 68], segmentation [95] and regression based methods such as global regression [25, 61, 43], local regression [27] and density estimation [66]. The approach we propose in this Chapter involves the use of a density estimation method, a well-known technique in the context of crowd counting [66], to count spikelets.

The task of counting wheat spikelets from infield images (as opposed to images obtained in some constrained lab environment) presents some real challenges because of their self-similarity, high volume per image, and severe occlusion as well as the challenges posed by lighting and other variations in the images captured.

Another challenge is that current methods for counting and related tasks in image analysis are often based on image processing or machine-learning approaches that require manual identification of features. This relies on expertise of the crop specialists and thus the feature engineering approach may differ from one problem to another.

An alternative powerful approach which we employ to avoid the feature identification problem is to apply deep learning to automatically extract useful features. Deep learning is based on learning data representations through the application of multi-layered neural networks, so features do not need to be identified beforehand. Also, it can lead to a high degree of accuracy in image classification tasks [64].

Therefore, our overall approach is as follows. We employ a fully convolutional model (SpikeCount) to perform density estimation from dot annotated images. We utilise additional labelled data for the density estimation by means of Transfer Learning. In addition, we investigate training SpikeCount with and without prior segmentation and compare the performance of each. Section 5.1 summarises the datasets we used, the architecture of SpikeCount, model optimisation and training procedure details. Section 5.2 summarises the early experimental results of this work [4]. Section 5.3 presents the full experimental setups of this work. Section 5.4 describes the performance results of testing SpikeCount and the their interpretation for each experimental setup. Also, we discuss results of testing SpikeCount for each growth stage. Finally, Section 5.5 presents our conclusions from this experiments.

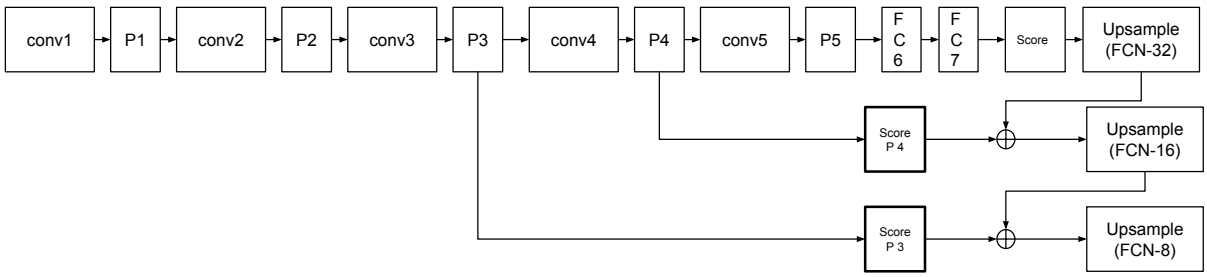

Figure 5.1: SpikeCount Architecture established for wheat spikelets density estimation regression.

5.1 Methods

In this section, we are going to explain SpikeCount architecture, the specification of model training including the cost function, the training hyperparameters, and the protocol of splitting the sequences into training, validation and test sets.

5.1.1 SpikeCount Architecture

In our approach, we apply a fully convolutional network to tackle the problem of spikelet counting. Fig. 5.1 represents our architecture. The last fully connected layers attached in any CNN-based classifiers are converted to convolutions. This ensures that the semantics of target objects are preserved which are essential for tasks that require structural predictions (predictions for each pixel), for example segmentation, because converting those layers to convolutions provide localisation and shape information about target objects. The FCN model can be adapted easily to tackle the issue of density estimation because (1) it is a task that also requires a structural target (pixels densities) (2) density estimation can benefit from recognising the semantics of target objects to be counted. Our model, SpikeCount, is composed of a Very Deep Convolutional Network (VGG16) [115] (Fig. 5.1: conv1-P5), formed by two fully convolutional (Fig. 5.1: FC6 and FC7) layers and three upsampling layers. The filter size selected for all convolutional layers is 3×3 with a stride of 1 and the max-pool layers have a pooling size of 2×2 with a stride of 2. We employ the concept of feature fusion by adding two skip connections (Fig.

5.1: after P3 and P4) to fuse the local features related to spikelets from lower layers to other shape and semantic features related to the wheat crops from higher layers. We added upsampling layers to ensure we recover the original image size affected by the application of repetitive convolutions and subsampling which reduces the input size.

5.1.2 Spike Region Segmentation

To investigate whether segmenting spike regions could enhance the spikelets counting task, we first manually remove the background using ground truth masks. In the next set of experiments, we use a CNN to tackle the segmentation automatically, according to our previous research, instead of a manual approach. We discuss further the experimental setup in detail in the experimental setup subsection.

5.1.3 SpikeCount Training

5.1.3.1 Cost Function

We found that using a pixel-wise $L2$ loss function for model optimisation gave the best results to regress the per pixel density (Equation 5.1) where $D^p_{GT_i}$ is the density ground truth and $D^p_{predicted}$ is the predicted density for a certain pixel p in image I_i.

$$L = \sum_{p \in I_i} (D^p_{GT_i} - D^p_{predicted})^2 \tag{5.1}$$

5.1.3.2 Training hyperparameters

Hyperparameters need to be initialised before the training process starts. Then, the training algorithm learns new parameters as part of the learning process [118]. Summary of the SpikeCount training hyperparameters values used in our study are as following:

- Weight θ (parameters) /Bias initialisation: It is good practice when training any deep learning model from scratch to initialise the weights with random values and the bias with 0. There are many initialisation techniques that can be used but we found He et al.'s [51] technique achieved the optimal results when training from scratch. This technique generates a mean centred normal distribution with standard deviation σ equal to $\sqrt{2/n_l}$ where n_l is the number of inputs in a certain layer l.

- Dropout rate probability: This parameter serves as a regulariser to reduce the model overfitting [118]. It determines how many units can be deactivated randomly for every training iteration in a certain layer. In our model, two dropout layers, with a value of 0.5 for p, which represents the probability of keeping the unit activation or not. This is recommended when using layers with large numbers of units [118] such as FC6 and FC7. Therefore, the dropout is implemented after every fully convolutional layer FC6 and FC7.

- Intermediate non-linearity unit: This is an essential component in any CNN that focuses on highlighting and emphasising the relevant features of the data and the task. As a default, we have selected Rectified Linear Unit (ReLu) for this parameter which is an element-wise thresholding operation that is applied on the output of the convolutional layer (resulting feature map) to suppress negative values: F(x)=max(0,x) where x is an element in the feature map.

- Epochs: This refers to the number of training iteration, and is different from one experimental setup to another.

- Optimisation algorithm: The weights are updated for every learning iteration using mini-batch RMSprop optimising algorithm [52], which we selected because it worked better with the density estimation regression problem we have, with a learning rate of 0.001. We have selected the initial learning rate by experimenting with lowest learning rate (1×10^{-5}) and linearly increasing

it until we reached the optima, i.e. when we reach the lowest validation loss for mini-batch of 20.

5.1.4 Training and validation of the architecture

We have selected the CQ_2015 sequence for training SpikeCount and the CQ_2016 sequence as the validation set to observe if there is overfitting of the model as explained in Chapter 3. However, because of the high resolution of the images (2592 $\times$ 1944), we randomly sampled 450 sub-images with a size of 512 $\times$ 512 with their corresponding manual labels. This approach is to reduce computational complexity, as dealing with the original image size would be unmanageable in the training process.

We have utilised an early stopping technique when training the model. Early stopping allows us to keep a record of the validation learning (e.g. cost and accuracy) for each learning epoch. It is a simple and inexpensive way to regularise the model and prevent overfitting as early as possible [63, 9]. We have selected the validation cost as the metric to observe for early stopping. The maximum epochs for observing the change in validation cost is 20 epochs. In other words, if the validation cost has not been decreased for 20 epochs, the model training will be stopped and the model weights resulting from the lowest validation cost are saved.

5.2 Spikelet Number Estimation - early work

In our early work [4], we had only 15 images available to us relating to the CQ_2016 dataset. We had a small number because of the limitations imposed by the task of dot annotating, which is time consuming and could therefore only be accomplished for a very reduced number of images initially. However, our early experimentation produced very promising results with good error rates, much improved by using both manual segmentation and Transfer Learning using ACID features. Therefore,

here we summarise the preliminary experiments we carried out and the results as reported in [4].

The application of Transfer Learning with ACID features is very important as data annotation required to extract 'ground truth' from images is expensive in terms of time and resources. Therefore, if we can leverage Transfer Learning in the task of density estimation [138, 90] to enable the use of already labelled images from other context this would be very beneficial.

In terms of training, we first formed the training and validation set from the ACID dataset according to the 80:20 split rule. This was performed in order to obtain some parameters using those ACID images that related to a similar task and therefore implement Transfer Learning. Then, we randomly sampled sub-images with a size of 512×512 for each set. After that, we manually selected 1241 sub-images from the training set and 303 sub-images from the validation set that contained spike regions. With those images we trained the model for 100 epochs for the Transfer Learning experiments described below.

As we only had 15 annotated images from the CQ_2016 dataset for these set of experiments, we decided to divide them into 3-folds for cross validation, with 5 images per fold. Then, we randomly subsampled 512×512 sub-images from each fold individually.

Moving to the preliminary results, Table 5.1 shows the cross-validated performance of the SpikeCount model for the different experimental set-ups. We show the effect of eliminating the background and other interfering noise by applying manual segmentation and we compare this with the values obtained without segmentation. In both cases, we test whether loading pre-trained ACID parameters has improved the SpikeCount performance for the task of counting spikelets. Table 5.1 shows that applying segmentation before counting has decreased the spikelet counting error to 82.2 and 102.00for MAE and MSE respectively when training SpikeCount from scratch. This represents a reduction of 83.5 % and 81.2% respectively for

MAE and MSE with respect to error measures without segmentation. In terms of Transfer Learning, loading ACID pre-trained parameters has a positive impact on the model performance by decreasing MAE and MSE to 53.00and 71.2 respectively when segmentation is also applied. This represents a decrease of 35.5% and 30.2% respectively when segmentation is applied with respect to the error from the scratch model. When no segmentation is applied, the Transfer Learning reduces error by 84.5% and 80.3% respectively for MAE and MSE. Hence, both segmentation and Transfer Learning have a very significant effect on error rates. It is worth noting that the pre-trained ACID parameters have minimised the gap between the SpikeCount performance with and without segmentation. The difference in missed spikelets when training from scratch is 415.8 for MAE and 441.5 for MSE. On the other hand, the comparative difference when loading pre-trained ACID parameters is 24.12 for MAE and 35.9 for MSE. Overall, the difference between the best model (with segmentation and Transfer Learning) and the worst (the scratch model without segmentation) is over 89% for MAE and over 86% for MSE.

Table 5.1: The Mean Absolute Error and Mean Squared Error of estimating the number of spikelets for two experimental setups (displayed as columns): training SpikeCount from scratch and by loading ACID dataset learned parameters and for pre-segmenting images (displayed in rows) on CQ_2016 images

	Scratch		ACID [90]	
	MAE	MSE	MAE	MSE
With Segmentation	82.20	102.00	53.00	71.20
Without Segmentation	498.00	543.50	77.12	107.10

5.3 Spikelet Number Estimation - full experimental set-up

In this section, we extend that work to a larger and more challenging set of images belonging to different growth stages which were acquired and dot annotated as the project progressed. In order to help understand and evaluate the impact of different CNN learning factors that can affect spikelets estimation, we have divided the experimentation into different experimental setups. For all experimental

setups, we test SpikeCount on three sets of CQ_2016 and CQ_2017: Original, Optimal and Prediction as shown in Figures 3.7 and 3.8 for 2016 and 2017 image sets respectively. The "Original set" contains images with a complete background, that is, no background is removed, which turned out to be the hardest problem as may be expected. The "Optimal set" refers to images with background removed according to the segmentation ground truth; this scenario enables us to measure the quality of spikelet counting when the background is eliminated completely so we can measure the effect of reducing the complexity of the problem. However, it would require quality segmentation annotations for each image, which may not be available in a "live" crop monitoring scenario. Finally, the "Prediction set" refers to the segmented images when using the prediction of the segmentation model proposed in Chapter 4. This can give us an assessment of the model performance of spikelets estimation relative to the segmentation quality of SpikeCount. It would provide for a completely automated model which could be employed in crop monitoring, but clearly overall quality will be affected by quality of the segmentation model. On the other hand, the removal of the background even if not perfect would reduce complexity in respect to the Original set. Testing on each set allows us to measure the model performance on specific scenarios.

Our experiments are as follows:

- **Experiment 1: Training SpikeCount from Scratch** - with these set of experiments we try to understand how well the architecture does when trained from scratch in a conventional way. We initialise the model weights with He et al.'s [51] technique as discussed in the SpikeCount Training subsection 5.1.3. We use both the Original images and the segmented images and compare performance.

- **Experiment 2: Training SpikeCount by loading ACID learnt weights** The aim of this experiment is to investigate whether high quality labelled wheat images (ACID images) that are similar to our images but captured

in more controlled environment can aid the model learning and improve the overall performance.

For the experiments reported in this Chapter, as in the previous experimentation, we first formed the training and validation set from the ACID dataset according to the 80:20 split rule. Then, we randomly sampled sub-images with a size of 512×512 for each set. After that, we manually selected 1241 sub-images from the training set and 303 sub-images from the validation set that contain spike regions. With those images we trained the model for 100 epochs for the Transfer Learning experiments. We then loaded parameters learned from training the model on the ACID dataset, as described earlier, and continued fine tuning the model using the CQ_2016 sequence Original images. We repeated experiments for segmented images.

- **Experiment 3: Investigating the importance of SpikeCount Encoder's Blocks when loading ACID learnt weights** - The aim of this experiments are to investigate which block(s) in SpikeCount encoder (VGG16) has the most influence in improving the model performance. There are five blocks in the encoder: conv1-conv5. In the previous experiment, we used ACID weights to aid in the spikelet counting problem by transferring knowledge from wheat images captured under controlled environments to our uncontrolled environment (i.e infield wheat crops). We initialised the model with the pre-trained ACID weights. In this experiment, not only do we initialise SpikeCount with ACID weights but we also freeze the model encoder's blocks gradually from conv1 to conv5 to investigate which ACID learnt features have the upmost positive effect on predicting spikelets density estimation and hence counting. Therefore, we will conduct this experiment as follows after loading ACID weights:

  1. Freeze (suspend training) the first block (conv1), and train the rest of model on CQ_2015; validate on CQ_2016 and test on CQ_2017.

2. Freeze (suspend training) the first and second blocks (conv1-2), and train the rest of model on CQ_2015; validate on CQ_2016 and test on CQ_2017.
3. Freeze (suspend training) the first three blocks (conv1-3), and train the rest of model on CQ_2015 ; validate on CQ_2016 and test on CQ_2017.
4. Freeze (suspend training) the first four blocks (conv1-4), and train the rest of model on CQ_2015; validate on CQ_2016 and test on CQ_2017.
5. Freeze (suspend training) the first five blocks (conv1-5), and train the rest of model on CQ_2015; validate on CQ_2016 and test on CQ_2017.

- **Experimental Setup 4: Investigate the importance of Augmentation** - Augmentation is one of the techniques used to reduce over-fitting and produce more generalised deep learning, particularly, when the data is limited [113]. We have employed a mix of basic image manipulation data augmentation that relies on geometric transformations on the fly on both images and density maps when training SpikeCount. Examples of data augmentation used are as follows:
  - Random vertical flipping.
  - Random colour space modification to decrease or increase brightness of a random value between 0.2 and 1.5.
  - Random rotation with a random value range between -10 degrees and 10 degrees.
  - Random translation by applying horizontal and vertical shift with a random value range between -0.6 and 0.6.
  - Random zoom augmentation with random value range between 50% and 150%.

We have repeated experiments 1 and 2 while employing augmentation.

5.4 Results

After training SpikeCount on CQ_2015 and validating it on CQ_2016 for all experimental setups discussed in the last section, in this section we assess the performance of the model for each experimental set using Original, Prediction and Optimal for both CQ_2016 and CQ_2017. For this, we use a sliding-window approach which is discussed in Chapter 3. We will analyse the results of spikelet counting in general and in terms of each growth stage as such breakdown may be important from a biological point of view. To analyse the results for each sequence, we need to take the average number of spikelets per growing season into context; these are 2027.50 spikelets for CQ_2016 and 2758.00 spikelets for CQ_2017. We also break down the counting results into background and spike regions for Original and Prediction sets to have a closer look at how the presence of background affects the model's performance.

5.4.1 Experiments 1 and 2: comparing training from scratch versus training with ACID learnt weights

We first evaluate the advantages of Transfer Learning using the additional images in the ACID dataset (Experiment 2) by comparing that to training from scratch (Experiment 1). Transferring knowledge from high quality labelled wheat images (ACID images) that are similar to our images but captured in more controlled environment can aid the model learning and improve the overall performance; the additional annotated images can enhance our training set. Tables 5.2 and 5.3 present the results of testing SpikeCount on CQ_2016 and CQ_2017 respectively. In both tables the results of training SpikeCount on CQ_2015 from scratch and loading ACID learnt weights are compared.

Original set

Table 5.2 corresponding to the 2016 images shows that for the Original set, the

model performed better (results highlighted in bold) without the Transfer Learning. When using Transfer Learning weights, the MSE increases by 30% with respect to the values without Transfer Learning and the MAE increases by 26.43%. On the other hand, SMAPE decreased in % value by 34 showing that for this particular measure performance is better with Transfer Learning. Because

Table 5.2: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for two experimental setups (displayed as rows): training SpikeCount from scratch and by loading ACID dataset learned parameters for the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on CQ_2016 images

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	**2709.92**	**2566.43**	89.43	1112.86	1015.39	33.58	1904.82	1789.72	64.46
Loading ACID Weights	3864.54	3488.43	**55.43**	**354.57**	**299.44**	**10.44**	**1718.22**	**1590.85**	**47.99**
Abs. Difference	1154.62	922.00	34.00	758.29	715.95	23.14	186.6	198.87	16.47
Improvement as %	29.99	26.43	-	68.14	70.51	-	9.80	11.11	-

Similar to CQ_2016 results, the CQ_2017 results in Table 5.3 illustrates that for the Original set the model performed better when trained from scratch on measures of MSE and MAE. The MSE decreased by 31.87% and the MAE decreased by 32.25% with respect to values with Transfer Learning. SMAPE, on the other hand, decreased by 33.4% when using Transfer Learning showing an improvement for this particular measure.

Table 5.3: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for two experimental setups (displayed as rows): training SpikeCount from scratch and by loading ACID dataset learned parameters for Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on CQ_2017 images

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	**3747.92**	**3414.08**	93.90	1877.36	1643.33	43.35	2738.24	2461.06	73.67
Loading ACID Weights	5501.65	5039.17	**60.46**	**876.56**	**658.15**	**18.00**	**2252.97**	**1993.42**	**54.49**
Abs. Difference	1753.73	1625.09	33.44	1000.8	985.18	25.35	485.27	467.64	19.18
Improvement as %	31.87	32.25	-	53.3	59.95	-	17.72	19.00	-

To better understand performance, Figures 5.2 and 5.3 break down the errors into those calculated from the spike regions and those calculated from the background regions. This is because the counts may be made of objects that were missed or overestimated in the spike region, but also from objects that were counted in the background region where the count should be zero. On aggregation those may

counteract each other so we present them separately for all experimental setups. We only do this for the Original and Prediction sets, since the Optimal set is supposed to remove all the background. For the Original data, both for the MSE (left graph) and MAE (right graph) the first column of the graphs (E1 corresponding to Experiment 1) show that when training from scratch the majority of errors are made on the spike (blue) region, though there are a few also in the background (red) region. However, for Experiment 2 (E2, second column of the graphs) when we apply Transfer Learning the majority of errors are attributable to the background region. Hence, for Original data we can see that the ACID features which contain a very constant background may have a negative impact which translates into maximising the errors on the background regions for our own images with a more complex background. That may explain why Transfer Learning has a negative effect on the Original images for both MSE and MAE measures. The ACID images aid with spike regions because they contain similar detail and features which explains the decrease in SMAPE , but they detract when it comes to counting on the background as they do not add anything for this, having a black background with no objects. This effect is not repeated in the Prediction set where the ratio of errors on the background and spikelet areas is more proportional in both experiments 1 and 2. Note that the Prediction set may have a reduced background area where the segmentation algorithm has not produced particularly good segmentation results, and that in itself would reduce background errors.

Figure 5.4 shows the correlation graphs of predicted spikelets number (within the spikes region only) using SpikeCount against actual numbers for all experimentation setups for Original set where the first column represents 2016 growing season and second column represents 2017 growing season. The first row represents the first (E1) and the second (E2) experimental results correlation. The second row represents the third (E3) experimental results correlation which will be discussed later. The x-axis represents the actual number of spikelets while the y-axis represents the predicted spikelets number. It is evident from the first row for both

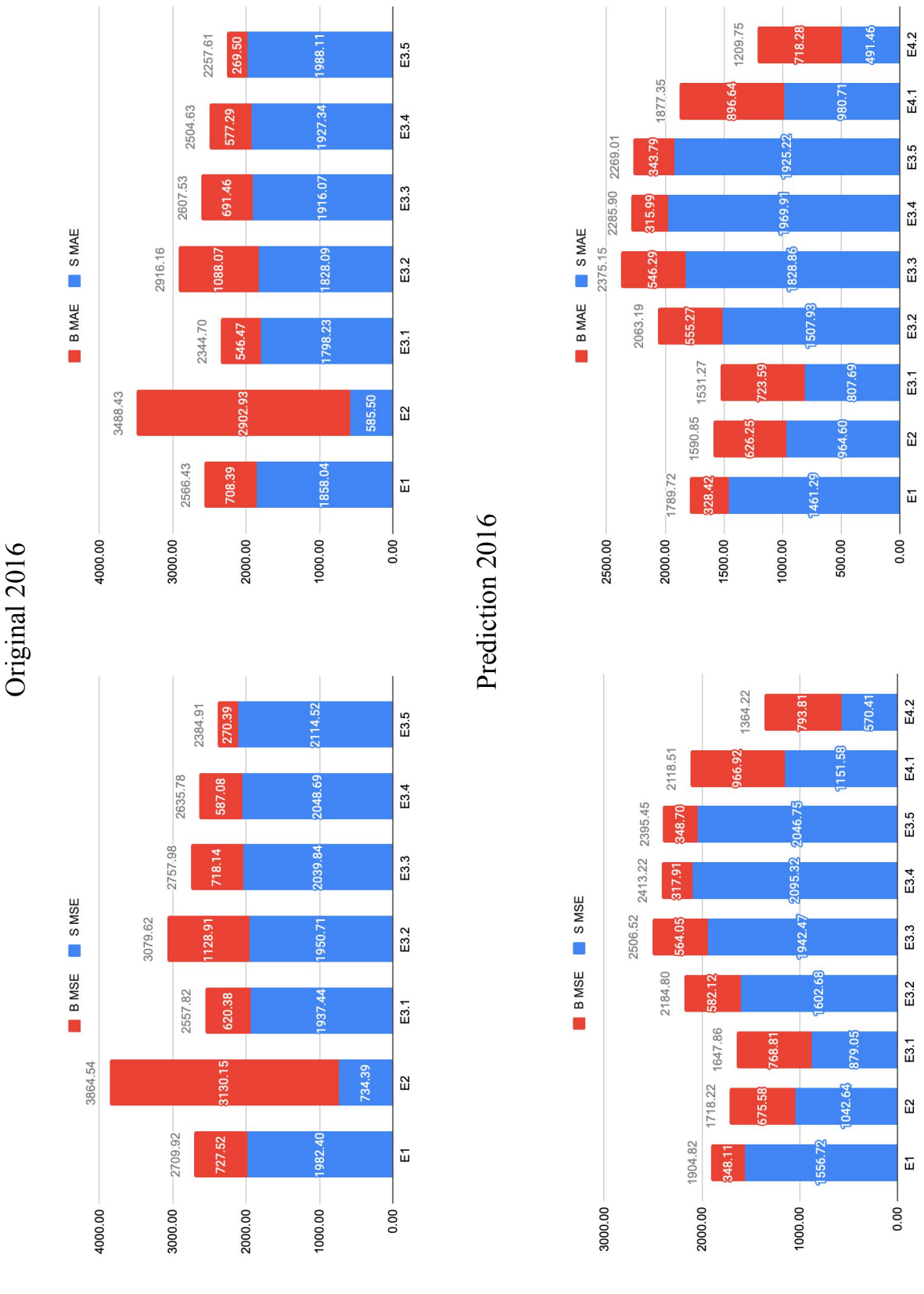

Figure 5.2: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of CQ_2016 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

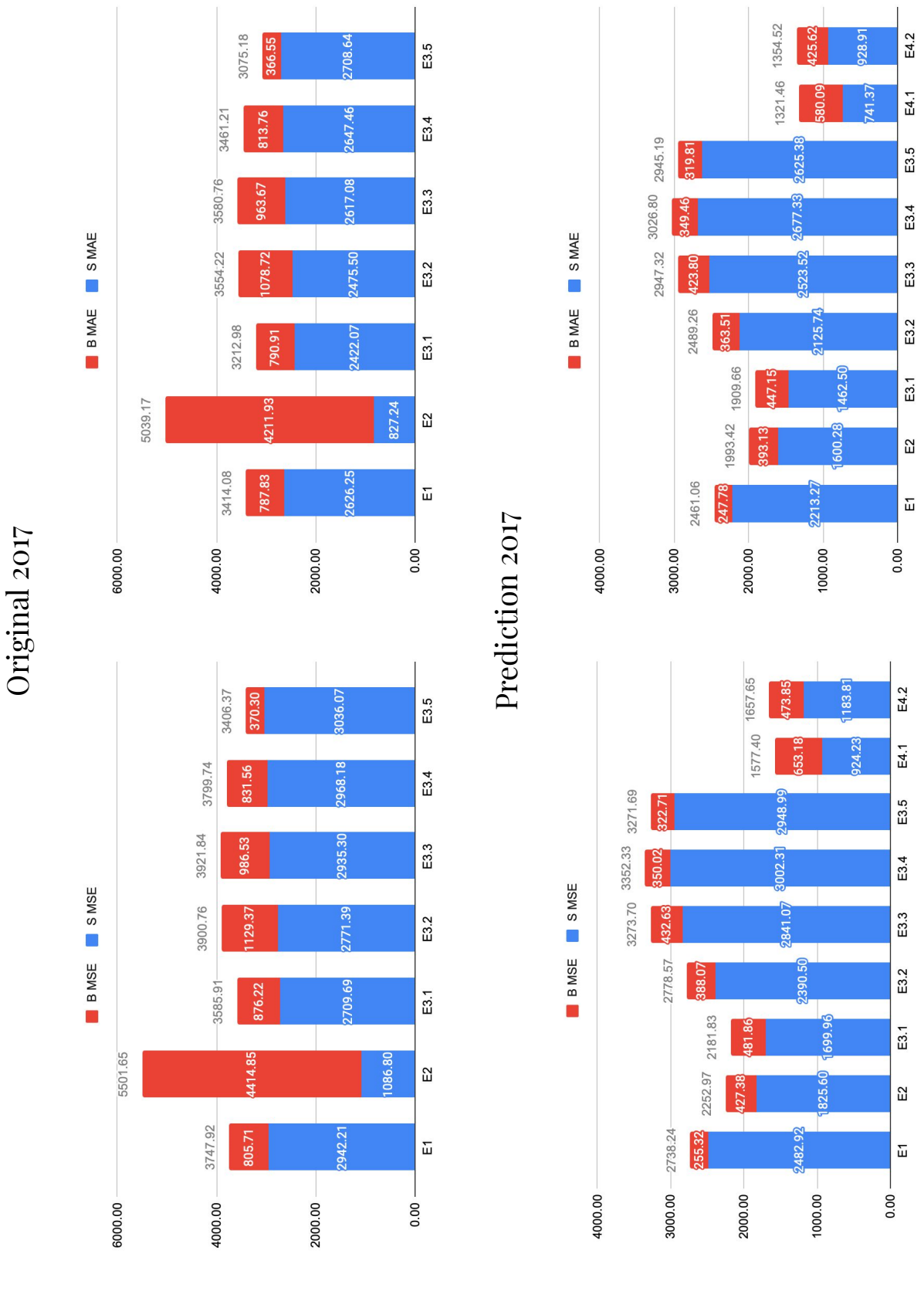

Figure 5.3: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of CQ_2017 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

CQ_2016 (left column) and CQ_2017 (right column) that initialising with ACID parameters result in good correlation with noticeable under-counting. However, as we stated above, for Original set and from the Figures 5.2 and 5.3, initialising ACID weights has resulted in severe spikelets over-counting in the background regions.

Contrary to results for the Original set, for both the Optimal and Prediction sets the performance improved markedly by using Transfer Learning.

Optimal set

For the Optimal set, according to Table 5.2 representing CQ_2016, the MSE, and MAE have decreased by 68.14%, 70.51% respectively with respect to values without Transfer Learning and there is also a decrease of 23.14% in SMAPE. Similar reductions of 53.3% for MSE, 59.95% for MAE and 25.35% for SMAPE are seen in Table 5.3 representing CQ_2017.

Figure 5.5 shows the correlation graphs of predicted spikelets number using SpikeCount against actual numbers for all experimental setups for Optimal set, where the first column represents 2016 growing season and second column represents 2017 growing season. The first row represents the first (E1) and the second (E2) experimental results correlation. The second row represents the third (E3) experimental results correlation. The third row represents the fourth (E4) experimental results correlation where both will be discussed later. The x-axis represents the actual number of spikelets while the y-axis represents the predicted spikelets number. It is clear from the first row for both CQ_2016 (left column) and CQ_2017 (right column) that initialising ACID weights has led to more correlated predicted spikelets numbers which show minimal under-counting.

Prediction set

For the Prediction set, according to Table 5.2 representing CQ_2016, the MSE and MAE have decreased by 9.8% and 11.11% respectively in relation to values without Transfer Learning, whereas SMAPE has reduced by 16.47%. For CQ_2017 in Table 5.3, the MSE, MAE and SMAPE have decreased by 17.72%, 19.00%, and 19.18%

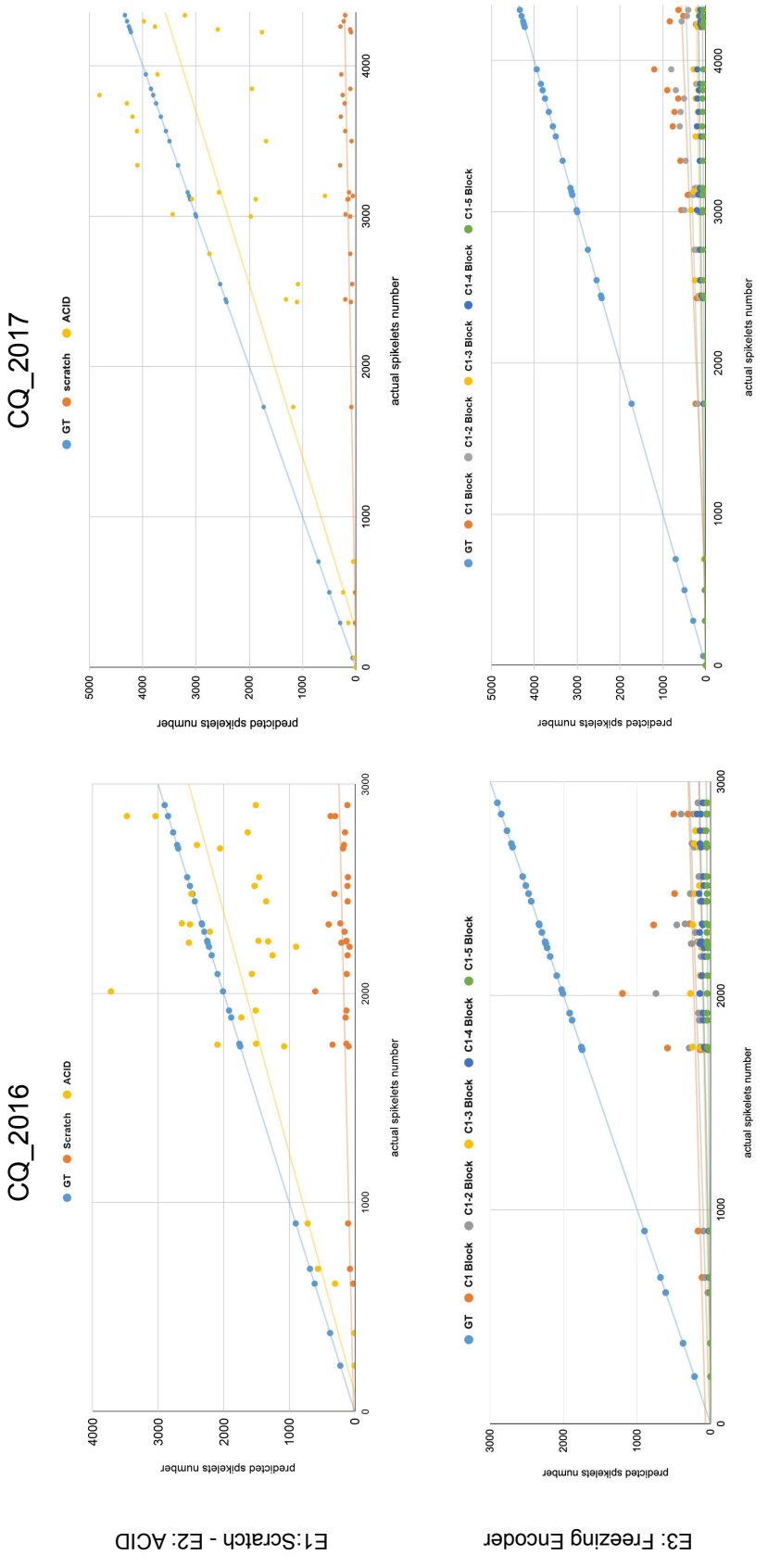

Figure 5.4: The correlation graphs of spikelets counting prediction excluding the background counting using SpikeCount and the ground truth counts for all experimentation setups for Original set; the first column represents 2016 growing season and second column represents 2017 growing season

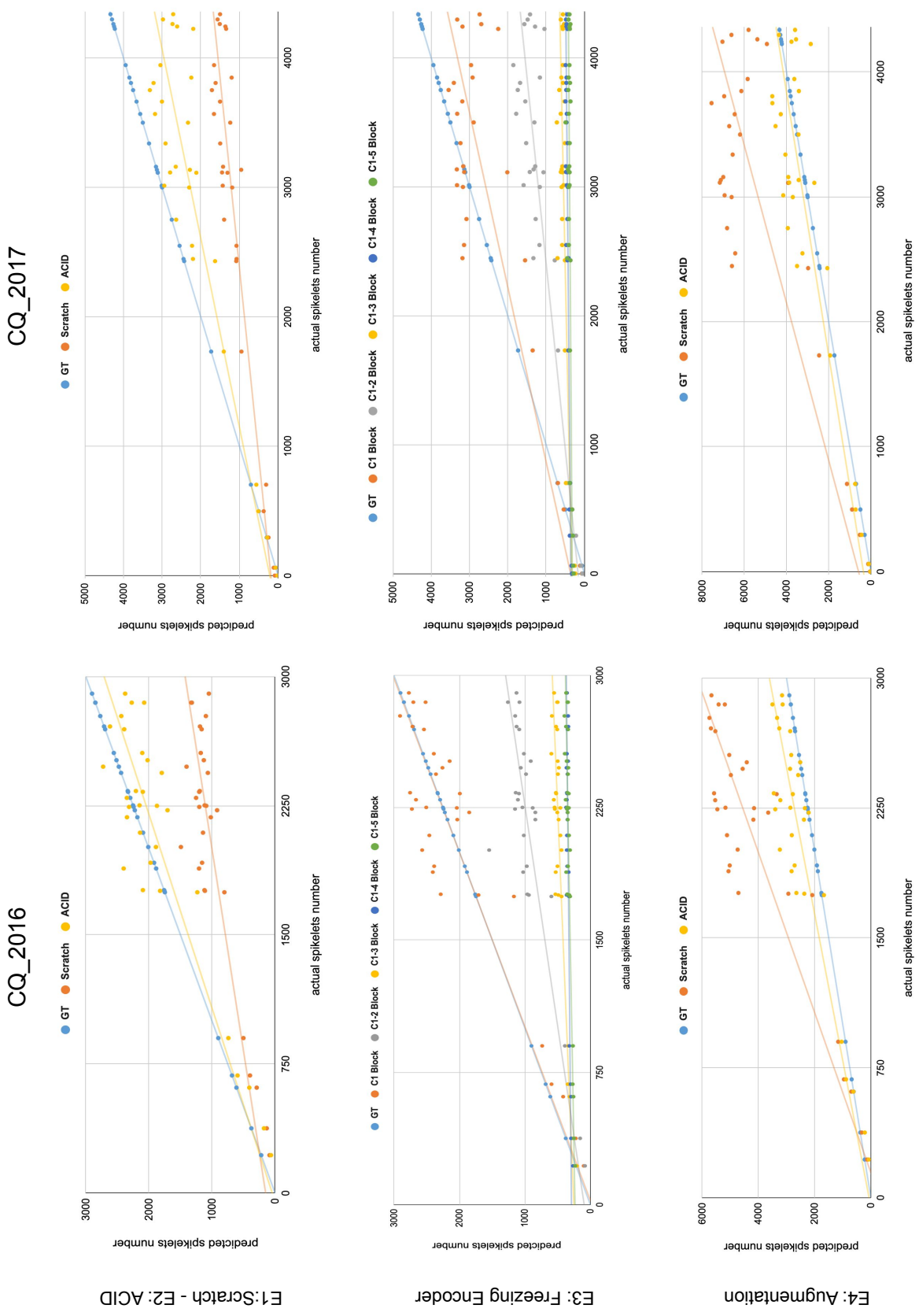

Figure 5.5: The correlation graphs of spikelets counting prediction excluding the background counting using SpikeCount and the ground truth counts for all experimentation setups for Optimal set; the first column represents 2016 growing season and second column represents 2017 growing season.

respectively. Those are more modest reductions than those seen in the Original or Optimal set.

Figure 5.6 shows the correlation graphs of predicted spikelets number (within the spikes region only) using SpikeCount against actual numbers for all experimentation setups for Prediction set where the first column represents 2016 growing season and second column represents 2017 growing season. The first row represents the first (E1) and the second (E2) experimental results correlation. The second row represents the third (E3) experimental results correlation. The third row represents the fourth (E4) experimental results correlation. It is evident from the first row for both CQ_2016 (left column) and CQ_2017 (right column) that initialising ACID weights has led to a less alignment predicted spikelet numbers which indicates under-counting.

Result comparison across all sets

Among the three scenarios, eliminating the background completely, as in the Optimal set, has resulted in the best performance for CQ_2016, only missing on average 354.57 spikelets per image for MSE and 299.44 spikelets for MAE and with a relative error of 10.44%. Eliminating the background with our model also produces some gains with respect to the Original images. For CQ_2017, eliminating the background completely has also resulted in the best performance with only missing on average of 876.6 per image for MSE and 658.2 for MAE and with a relative percentage error of 18%. It is clear that focusing the counting on the ROI is important to obtain more accurate counting results.

It is clear also that for both CQ_2016 and CQ_2017 removing the background completely as it is the case with the Optimal set allows the features learned from the ACID dataset through Transfer Learning to have the most marked impact. There is also a positive impact on the Prediction set but not as marked.

Some detailed visual analysis, in this case for the ACID images, is shown in Fig. 5.7. By comparing the density maps generated from our models (column (b)) of

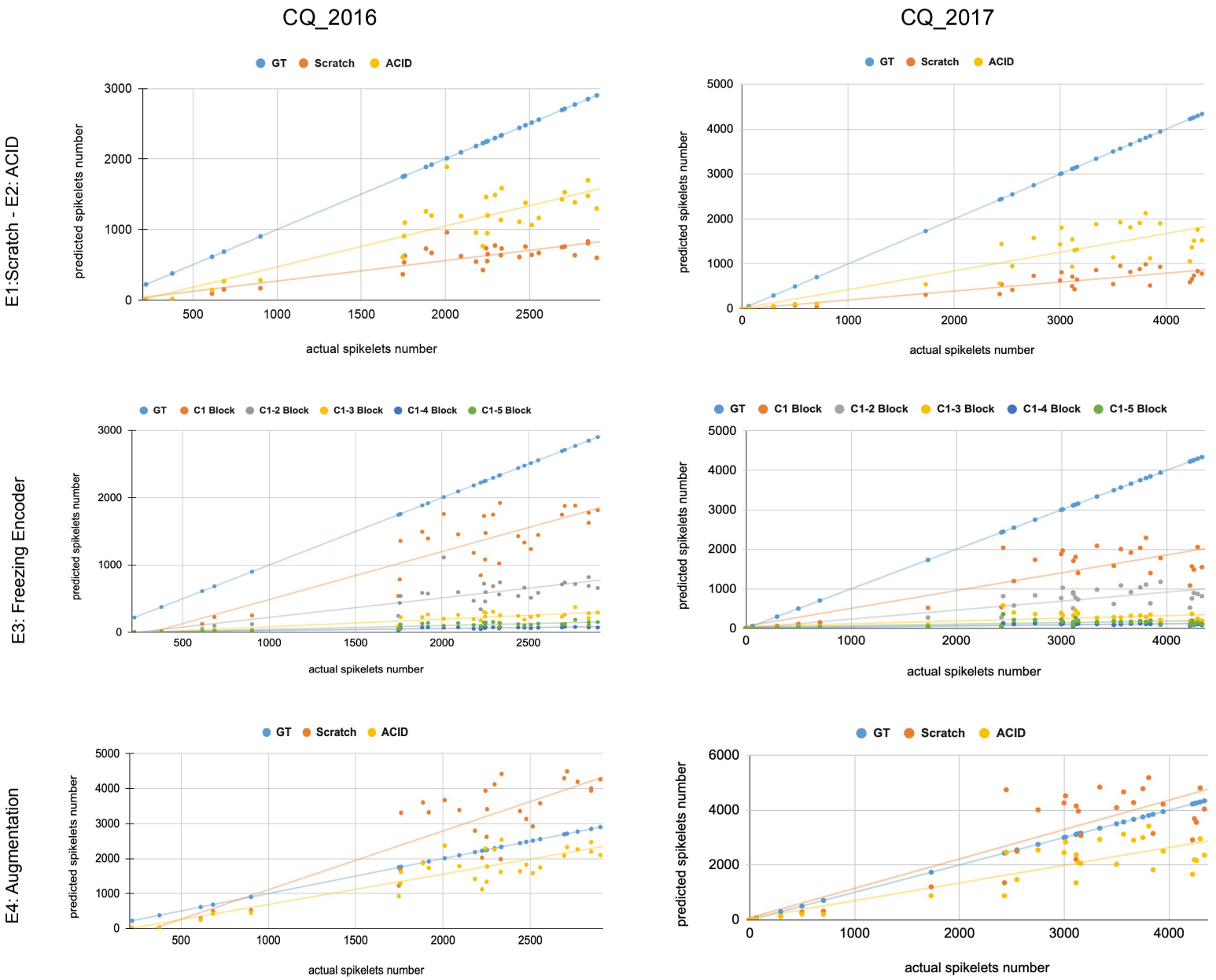

Figure 5.6: The correlation graphs of spikelets counting prediction excluding the background counting using SpikeCount and the ground truth counts for all experimentation setups for Prediction set; the first column represents 2016 growing season and second column represents 2017 growing season.

Fig. 5.7 with the 'ground truth' density maps derived from the dot annotation (column (c)) we can note that in some images, SpikeCount may be considered as over-counting because it is able to detect spikelets that were miss-annotated (missed) by accident in the dot annotation. For example, in Fig 5.7, SpikeCount predicted spikelet number for the second and third images as 53.47 and 49.93 while the ground truth for both images was 46.00 and 46.61. However, in these images, spikes that appear to contain a single row of spikelets in the dot annotation are recognised as having more spikelets by the SpikeCount model and this seems to correlate to the images in column (a). We can assume that as the dot annotation gets much more complex in the very crowded infield images, dot annotation may also be more inaccurate, so some of our errors may reflect the inaccuracies of our ground truth.

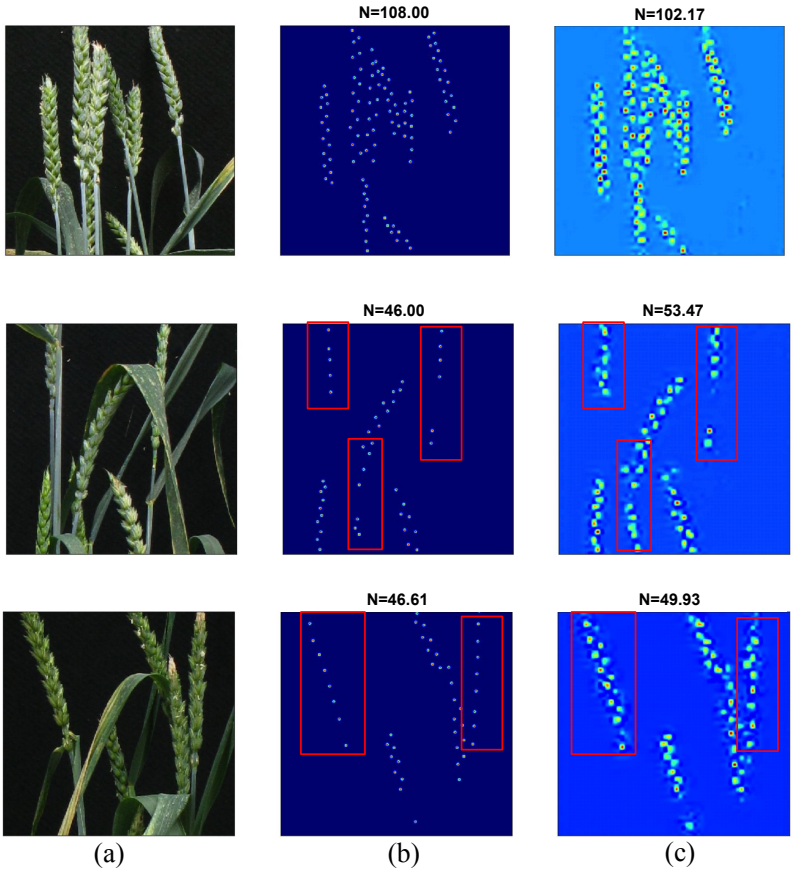

Figure 5.7: Visualisation of density maps resulting from testing for the Adapted SpikeCount on ACID dataset, where (a) represents image patch, (b) represents 'ground truth' spikelet density map and count obtained after dot annotation, and (c) represents the predicted spikelet density map and count.

5.4.2 Experiment 3: investigating the importance of SpikeCount encoder blocks

Next we investigate which ACID hierarchy features represented by the convolutional blocks in the SpikeCount encoder (VGG16) have the most important impact in estimating spikelets. This is done by gradually freezing the model encoder's block after loading ACID learnt weights. We then proceed with training and then we conduct the evaluation.

When evaluating improvement we do this against our best results for our previous experimental phase which are also reported in the Tables 5.2 and 5.3 to aid comparison, that is, against the MSE and MAE obtained by training from scratch for the Original set and the Transfer Learning results for all other measures and scenarios. We aim to establish if the new results obtained by freezing specific layers for a particular set, represent an improvement over the previous result for that set. Tables 5.4 and 5.5 present the results for both CQ_2016 and CQ_2017 respectively.

Table 5.4: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for the third experimental setup that investigates which feature hierarchy has the most effect on improving the results when loading ACID dataset learned parameters (displayed as rows) for Original, pre-segmented (Optimal) and pre-segmented (predicted) images (displayed in columns) on CQ_2016 images where numbers in italic are selected as the best results

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Freezing Conv1 Block	*2557.82*	*2344.70*	86.10	**315.96**	**270.29**	**8.87**	**1647.86**	**1531.27**	**44.65**
Freezing Conv2 Block	3079.62	2916.16	89.16	1216.558	1121.781	38.435	2184.802	2063.195	69.733
Freezing Conv3 Block	2757.982	2607.525	92.835	1685.330	1552.151	57.214	2506.520	2375.151	86.856
Freezing Conv4 Block	2635.776	2504.633	93.198	1831.979	1688.807	65.815	2413.221	2285.898	95.545
Freezing Conv5 Block	**2384.91**	**2257.61**	96.895	1833.346	1694.937	66.604	2395.447	2269.008	92.399
Best from Exp 1/2	2709.92	2566.43	**55.43**	354.57	299.44	10.44	1718.22	1590.85	47.99
Absolute Diff.	324.31	308.82	30.67	38.61	29.15	1.57	70.36	59.58	3.34
Improvement as %	11.97	12.03	-	10.89	9.73	-	4.09	3.75	-

Original Set

For the Original set, Table 5.4 shows that freezing the whole encoder (conv1-5) has decreased the MSE and MAE with respect to the previous experiments by 11.97 % and 12.03% and gives the best results. However, this could be misleading for two reasons explained below.

Table 5.5: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for the third experimental setup that investigate which features hierarchy has the most effect on improving the results when loading ACID dataset learned parameters (displayed as rows)for Original, pre-segmenting (Optimal) and pre-segmenting (predicted) images (displayed in columns) on CQ_2017 images

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Freezing Conv1 Block	*3585.91*	*3212.98*	86.44	**752.53**	**536.86**	**16.53**	**2181.83**	**1909.66**	**51.35**
Freezing Conv2 Block	3900.76	3554.22	88.48	1887.90	1646.39	43.38	2778.57	2489.26	71.62
Freezing Conv3 Block	3921.84	3580.76	93.60	2579.92	2276.80	65.84	3273.70	2947.32	88.03
Freezing Conv4 Block	3799.74	3461.21	94.72	2686.52	2388.34	71.14	3352.33	3026.80	95.54
Freezing Conv5 Block	**3406.37**	**3075.18**	97.26	2740.11	2430.58	73.09	3271.69	2945.19	92.72
Best from Exp 1/2	3747.92	3414.08	**60.46**	876.56	658.15	18.00	2252.97	1993.42	54.49
Absolute Diff.	341.55	338.82	25.98	124.03	121.29	1.47	71.14	83.76	3.14
Improvement as %	9.11	9.92	-	14.15	18.43	-	3.15	4.20	-

We go back to Fig. 5.2 where we presented breakdown of errors from the spike and background regions, and we focus on the Original set (top row) and columns E3.1 to E3.5. It is clear that suspending the training of the whole encoder (column E3.5 in the graph) has minimised the errors calculated for the background regions which indicates that the model is not estimating spikelets in those background regions. This may not be because of efficacy of the model but instead because the model was not able to learn in reasonable manner due to freezing the model encoder (conv1-5). Additionally, Fig. 5.2, Original set (top row), also shows that the spike region errors (blue areas in the bars) have not improved but, on the contrary to background region errors, they have increased the more we suspend blocks. Moreover, Figure 5.4, Original set-CQ_2016 (bottom left graph) presenting the correlation graphs, clearly illustrates that all variants of freezing the model encoder except for the first conv1 block (orange dots) have led to severe under-counting compared to the ground truth (blue dots).

These two aspects indicate that freezing two or more encoder block(s) hinders the model learning when training on images with full background such as the Original set. This could be explained by the differences between images in the Original set and the ACID set which is captured in a background free environment. Therefore, we will ignore the results of E3.2/3/4/5 as not particularly valid. So in other words, freezing only conv1 block has led to what we consider to be the most useful results and decreased the MSE and MAE with respect to the previous experiments by 5.61

% and 8.64% (Table 5.4) which is represented in italic)

Table 5.5 for the CQ_2017 images together with Figure 5.3, Original set(top row), and Figure 5.4, Original set (bottom right graph) confirm the same results and trends.

Optimal set

Table 5.4 shows that loading the ACID weights and suspending the training of the top convolutional block (conv1) improved the results for Optimal and Prediction sets of CQ_2016 sequence relatively to previous experiments (Table 5.2). For the Optimal set, the MSE, MAE and SMAPE have decreased by 10.89%, 9.73% and 1.57%.

Table 5.5 illustrates similar results for CQ_2017. Loading the ACID weights and suspending the training of the top convolutional block (conv1) improved the results for the Optimal set in relation to previous experiments. We observe that MSE, MAE and SMAPE have decreased by 14.15%, 18.43% and 1.47% for the Optimal set.

It is clear from Figure 5.5 showing the correlation graphs, that for the Optimal set CQ_2016 (leftmost graph in the second row) and Optimal set-CQ_2017 (rightmost graph in the second row) the spikelet predictions resulting from freezing conv1 block after loading ACID parameters are the most alignment with the ground truth counts.

For both Optimal (Fig. 5.5) and Prediction sets (Fig. 5.6), it is also clear that the predicted counts deteriorate the more we freeze blocks.

Prediction set

The Prediction set follows from the Optimal set but with more modest improvements. For CQ_2016 presented in Table 5.4, loading the ACID weights and suspending the training of the top convolutional block (conv1) has resulted in decreases of 4.09%, 3.75% and 3.34 % for MSE, MAE and SMAPE respectively. For

CQ_2017 (Table 5.5) MSE, MAE and SMAPE have decreased by 3.15%, 4.20% and 3.14 %.

Also, from the correlation graphs in Figure 5.6 (second row) for both CQ_2016 and 2017, the spikelet predictions resulting from freezing conv1 block after loading ACID parameters show noticeable under-counting. It is also clear that the predicted counts deteriorate the more we freeze blocks.

Result comparison across all sets

As a summary for this experiment, it is clear that freezing the first block encoder (conv1 block) can be beneficial in improving the result of estimating spikelets in the Original, Optimal and Prediction. As the features at the top of the hierarchy layers are general and independent from the target dataset, hence, preserving those by not training the top layer can lead to better results. However, the layers (conv2-5) have features that are more dependent on the datasets, so freezing these layers can inhibit the learning process leading to worse results.

For both CQ_16/17 the error in the spike regions for both Original and Prediction sets are increasing the more encoder blocks we suspend, which indicates that it is important to train the convolutional layers (conv2-5) that are related to the target dataset and only freeze lower layers such as Conv1 as those could be universal and represent the shared low level features.

5.4.3 Experiment 4: Investigating the importance of data augmentation

Lastly, we investigate how using augmentation could improve the task of spikelet counting. We have applied a simple mix of basic image manipulations data augmentations that rely on geometric transformations such as random rotation, translation, vertical flipping and brightness when training SpikeCount. Table 5.6 summarise the quantitative results of segmentation performance for the 2016 and 2017 dataset when training SpikeSEG.

Tables 5.7 and 5.8 present SpikeCount performance results of image augmentation when applied to both Experiment 1 and Experiment 2 (i.e. training from scratch and with Transfer Learning from ACID) in case with either approach data augmentation could produce real improvements. We tested the models on Optimal and Prediction sets since both gave promising results from previous experiments, but we do not test on the Original images for these experiments as those were very sub-optimal results. We compare against the best results from Experiment 2, since both Optimal and predicted sets had the Transfer Learning producing the best results.

Table 5.6: Quantitative Results of Segmentation performance for the 2016 and 2017 dataset when training SpikeSEG showing different evaluation scores where GA refers to Global Accuracy, MA refers to Mean Accuracy and MIoU refers to Mean Intersection over Union.

Dataset	GA	MA	Spike Accuracy	MIoU	Spike IoU
2016	93.34%	86.11%	75.85%	77.63%	62.75%
2017	92.47%	79.48%	60.7%	73.52%	55.35%

Table 5.7: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for two experimental setups utilising augmentation (displayed as rows): training SpikeCount from scratch and by loading ACID dataset learned parameters for Original, ground truth segmented (Optimal) and model segmented (predicted) images (displayed in columns) on CQ_2016 images

	Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training from scratch	2057.30	2333.99	30.27	2118.51	1877.35	38.08
Loading ACID Weights	459.75	587.77	11.70	**1364.22**	**1209.75**	**34.24**
Best from Exp 2	**354.57**	**299.44**	**10.44**	1718.22	1590.85	47.99
Absolute Diff.	105.18	288.33	1.3	354	381.1	13.75
Improvement as %	22.88	49.05	-	25.95	31.50	-

Table 5.8: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for two experimental setups utilising augmentation (displayed as rows): training SpikeCount from scratch and by loading ACID dataset learned parameters for Original, pre-segmenting (Optimal) and pre-segmenting (predicted) images (displayed in columns) on CQ_2017 images

	Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training from scratch	2207.85	2658.94	32.74	**1577.40**	**1321.46**	**31.90**
Loading ACID Weights	**667.34**	**544.32**	**16.32**	1657.65	1354.52	37.74
Best from Exp 2	876.56	658.15	18.00	2252.97	1993.42	54.49
Absolute Diff.	209.22	113.83	1.68	931.51	638.9	22.59
Improvement as %	31.35	20.91	-	70.59	47.17	-

Optimal set

The CQ_2016 results in Table 5.7 show that it is still Transfer Learning from the previous experiments that produces the best result for all measures in the Optimal set. The data augmentation produces greater (worse) error metrics than Transfer Learning alone with an increase in error metrics of 22.88% 49.05% and 1.3% for MSE, MAE and SMAPE respectively.

Training from scratch with data augmentation produces some improvements over training from scratch without using data augmentation (comparative data shown in Table 5.2) but not sufficient to advance on the best results obtained with Transfer Learning in Experiment 2.

On the other hand, the CQ_2017 results shown in Table 5.8 indicate a more positive evaluation. Data augmentation seems to have produced improved results in the Optimal dataset in conjunction with Transfer Learning, reducing the error by 31.35%, 20.91% and 1.68% for MSE, MAE and SMAPE respectively in relation to Transfer Learning alone.

The correlation graphs in Figure 5.5 (leftmost graph in the third row for 2016 and rightmost graph for 2017) show that for the Optimal set, the spikelet predictions resulting from using augmentation after loading ACID parameters (yellow dots) show a good correlation with slight over-counting, while only using augmentation (orange dots) while training from scratch leads to more over-counting.

Prediction set

According to Table 5.7, there is some improvement in adding data augmentation to Transfer Learning for the Prediction set with respect to the results of Experiment 2, with error decreasing by 25.95%, 31.50%, and 13.75% for MSE, MAE and SMAPE respectively.

Again for CQ_2017, the results shown in Table 5.8 indicate that data augmentation has improved the results, more significantly for 2017, in conjunction with training from scratch for MSE by 70.59%, MAE by 47.17% and SMAPE by 22.59%.

The correlation graphs for the Predication set in Figure 5.6 (leftmost graph in the third row for 2016 and right most graph for 2017) show that the spikelet predictions resulting from using augmentation after loading ACID parameters (yellow dots) result in good correlation with slight under-counting while only utilise augmentation (orange dots), leads to over-counting.

Result comparison across all sets

In summary for these experiments, data augmentation can provide some improvements for the prediction set and in some cases also for the Optimal set, particularly for the more challenging CQ_2017 images. We did not try data augmentation on the Original set.

5.4.4 Phenotypic analysis Spikelets Estimation for Growth Stages

In this section, we analyse spikelets estimation results from the perspective of wheat growth stages across all experimental setups. This is important from a biological perspective as growth stages may present very different images with varying numbers of spikelets. As mentioned in Chapter 3, the wheat images for CQ_2015, CQ_2016 and CQ_2017 were sampled from four different growth stages: booting, Heading, Flowering, and Grain filling. To analyse the results for each growth stage, we need to take the average number of spikelets per growth stage and the number of images into the context. It is worth noting that in regards to reporting the results of combined errors, although the figures presented may included Experiment 3 variation (E3.2-E3.5) for the reasons we highlighted in Subsection 5.4.2 we do not discuss these results in this section. To recap, freezing two or more encoder blocks has hindered the model's ability to learn the task in hand.

(a) Late Booting: GS47 **(b)** Late Booting: GS49 **(c)** start of Heading: GS51 **(d)** start of Heading: GS57

Figure 5.8: The sequence of spike(ear) emergence at two growth stage Booting and Heading [82, 85]

5.4.4.1 G40-49: Booting

According to the Zadok's growth decimal scale [141], there are ten major cereal growth stages with each growth stage divided into ten sub-stages [85]. Booting (G40-49) is the fifth growth stage on the Zadok's growth decimal scale. It starts after the stem elongation (G30-39). Booting growth stage starts with visibility of flag leaf sheath (the uppermost leaf on the stem) which contains the spike (ear) inside. So while the ear remains protected by the leaf sheath the stage is booting [82]. Initially, a lower number of spikes may be visible but in late booting they become more partially visible as shown in Figure 5.8.

There are only two wheat images in the booting growth stage in CQ_2016 with an average of 298 spikelets between them as shown in Figure 5.9.

On the other hand CQ_2017 has five booting images. The last three images of the sequence have an average of 259 spikelets between them as shown in Figure 5.10.

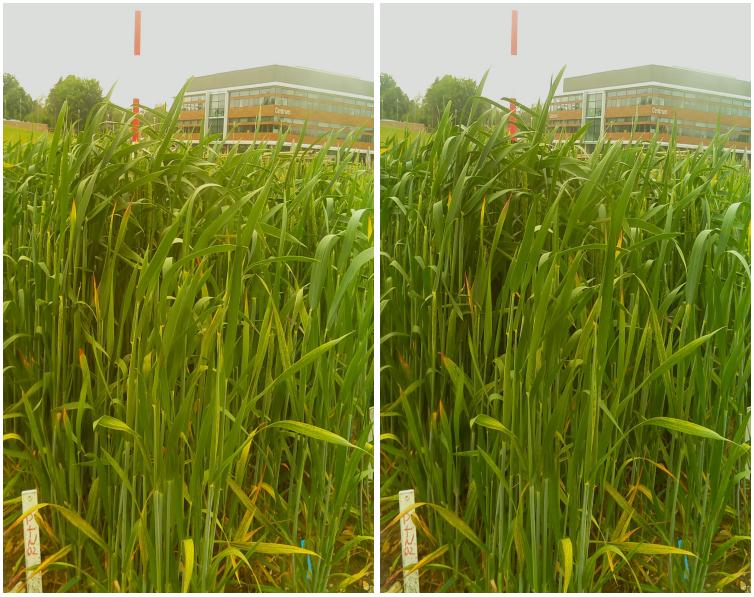

Figure 5.9: Booting growth stage images of CQ_2016

However, for this early growth stage there are no apparent spikes in the first two images of CQ_2017. On inspection, the wheat scenes captured at this growth stage are complex as they are overwhelmed with noisy background objects (i.e leaves) similar in colour and texture to the not so many undeveloped and partially hidden spikes. Therefore, this growth stage will be challenging particularly when testing Original sets.

Table 5.9 for CQ_2016 and Table 5.10 for CQ_2017 report on the various experimental setups for three sets: Original, Optimal and Prediction. In addition, Figures 5.11 and 5.12 provide the quantitative errors calculated in relation to estimated spikelets from the spike and background regions separately for the booting growth stage in CQ_2016 and CQ_2017 respectively. Also, Figures 5.13 and 5.14 show visualisation of the spikelet density maps for the Booting stage of 2016/2017 image series.

Original Set

First, if we look at the overall results for the Original set, according to Tables 5.9 for CQ_2016 freezing of the first convolutional block (E3.1) results in the best values of MSE and MAE. On the other hand, from Table 5.10 for CQ_2017 training the

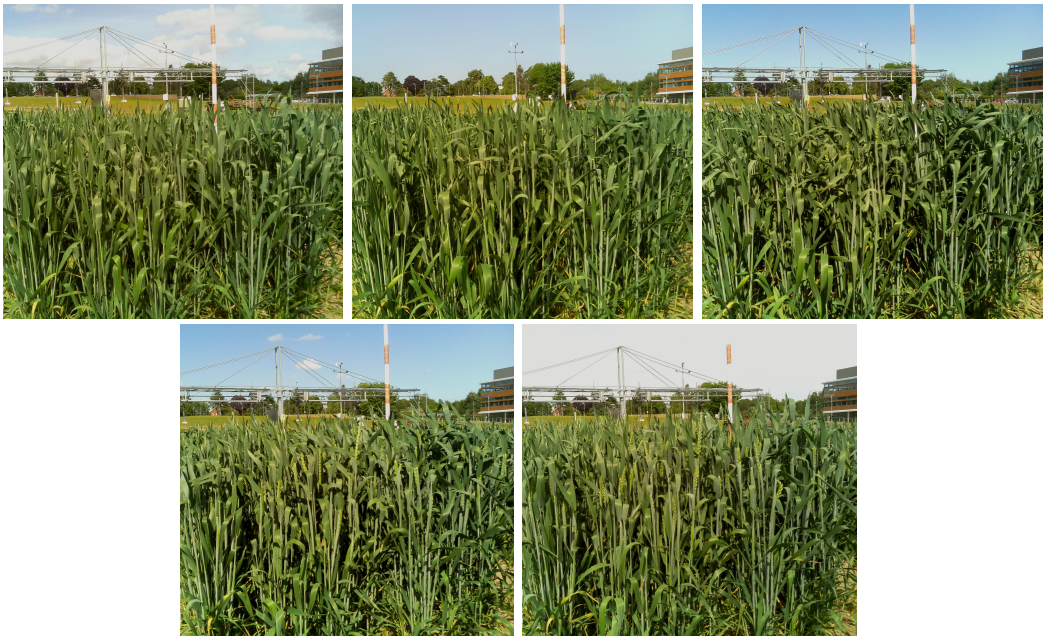

Figure 5.10: Booting growth stage images of CQ_2017

Table 5.9: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Booting** growth stage of CQ_2016 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	673.17	662.24	99.80	195.98	187.88	45.42	**555.07**	**543.19**	98.55
Loading ACID Weights	1942.26	1925.73	**99.67**	179.06	177.64	45.37	741.20	719.47	97.13
Freezing Conv1 Block	**588.37**	**576.78**	99.92	136.81	135.71	31.18	817.59	794.43	97.09
Augmentation-Scratch	-	-	-	**70.68**	**69.13**	**15.67**	964.15	936.96	**95.10**
Augmentation-ACID	-	-	-	154.06	154.25	38.60	820.44	797.38	95.64

model from scratch (E1) results in the best values of MSE and MAE. Breaking the errors into those on the background and spikelet areas, as Figures 5.11 and 5.12 show, we can see that for CQ_2016 there is no significant change in errors made in spikelet areas (range between 303 to 307 for MSE and 294 to 297 for MAE) across all experiments. This may be because all model variations find it challenging to detect and count the very few spikelets present which leads to extreme under-counting. However, when we measure MSE and MAE within background areas, we find the measures change significantly from one experiment to another. For example, the highest measured MSE and MAE are 1638.27 and 1631.23 for Transfer Learning using ACID weights (E2) hence, Transfer Learning alone increases errors on the

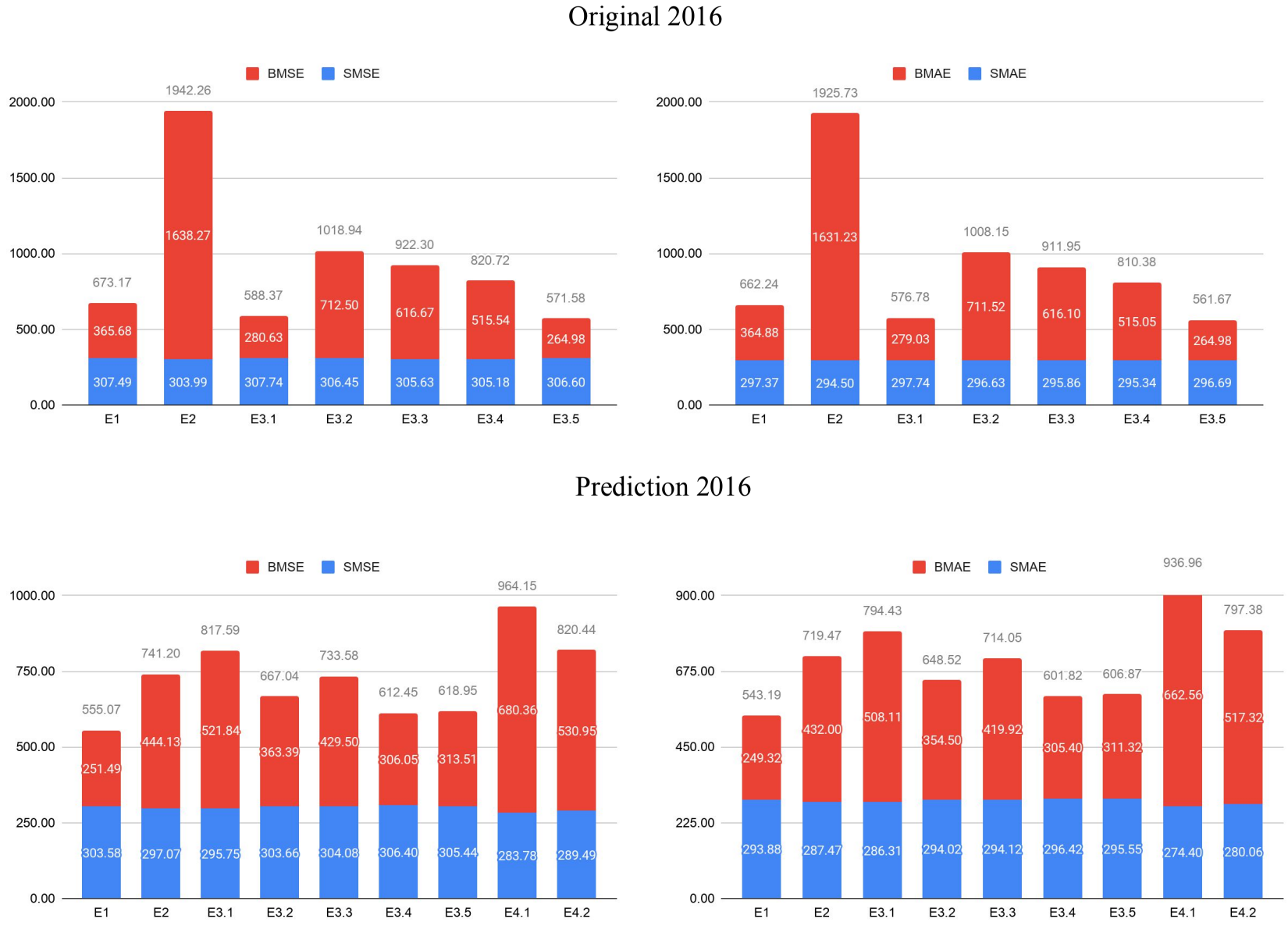

Figure 5.11: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Booting** growth stage-CQ_2016 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

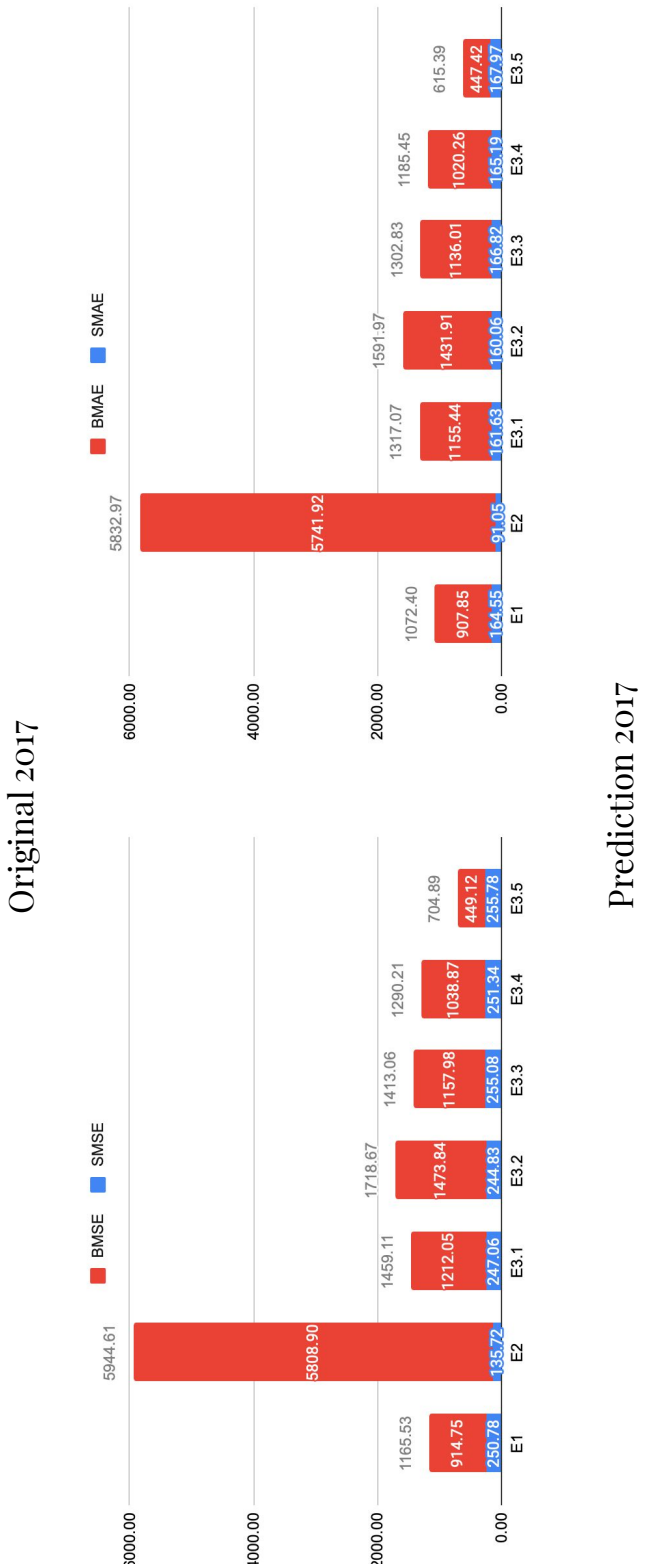

Figure 5.12: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Booting** growth stage-CQ_2017 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

Table 5.10: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Booting** growth stage of CQ_2017 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	**1165.53**	**1072.40**	99.14	82.57	77.73	50.55	407.63	329.50	92.39
Loading ACID Weights	5944.61	5832.97	**97.56**	**12.34**	**11.61**	**41.28**	405.26	330.83	89.02
Freezing Conv1 Blocks	1459.11	1317.07	98.97	50.64	49.46	47.65	458.86	385.48	89.54
Augmentation-Scratch	-	-	-	206.94	152.48	58.20	431.42	382.61	**80.10**
Augmentation-ACID	-	-	-	119.88	87.30	51.88	**386.92**	**327.46**	81.26

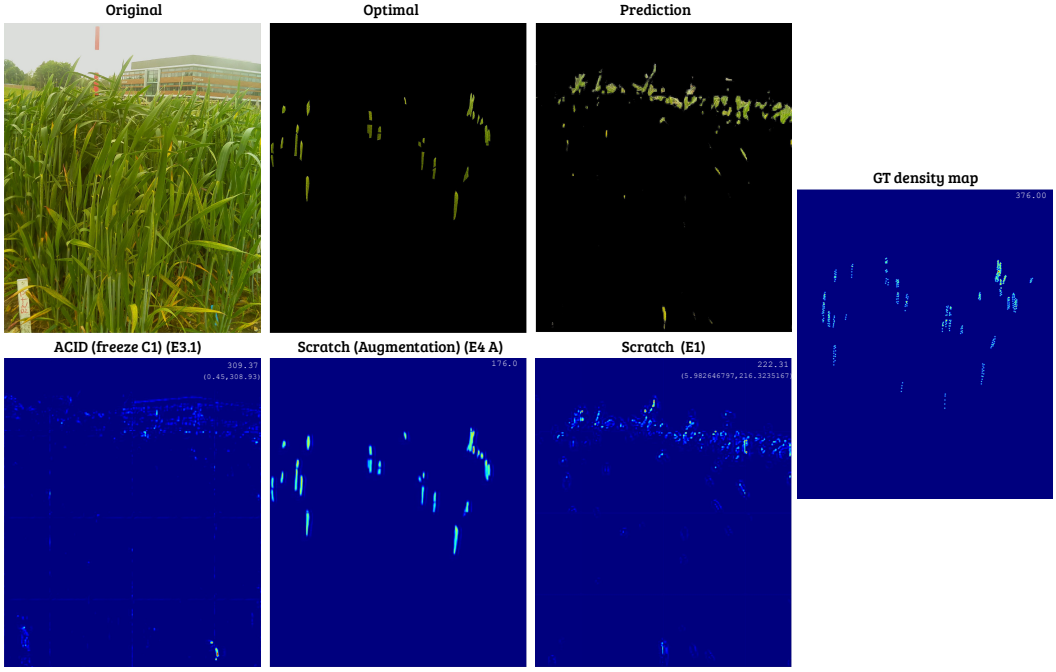

Figure 5.13: Visualisation of the spikelets density maps for the **Booting** stage of 2016 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction image while the second row shows the best resulting density map with predicted spikelets number(top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

background areas. On the other hand, suspending the training of the the first encoder after SpikeCount is initialised with ACID weights results in improving the model's ability to ignore what may appear like spikelets in the background with MSE and MAE of 280.63 and 279.03 respectively. That explains why freezing the first encoder after initialising the model with ACID weights results in the best outcome for the combined MSE (588.37) and MAE (576.76). The SMAPE measure gives best value when loading ACID weights (E2).

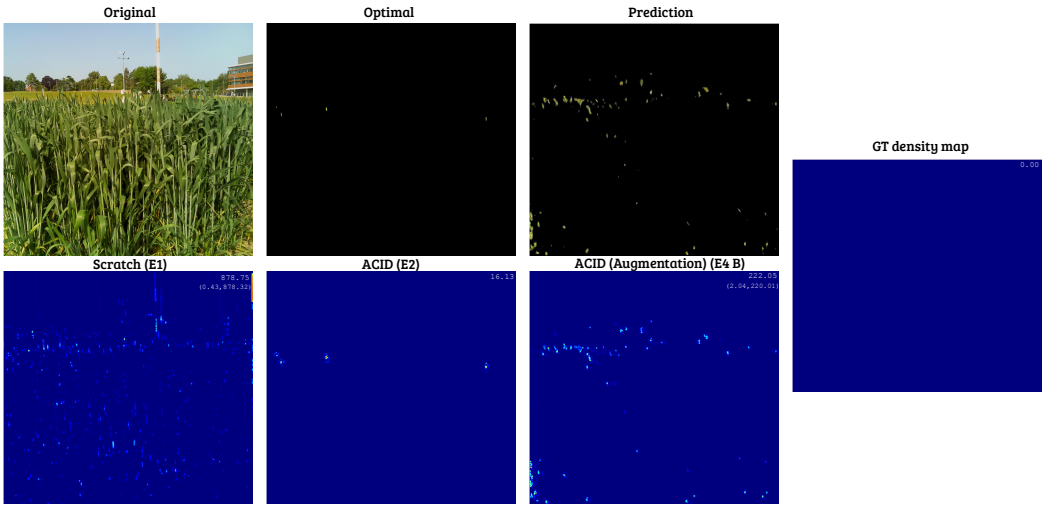

Figure 5.14: Visualisation of the spikelets density maps for the **Booting** stage of 2017 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction image while the second row shows the best resulting density map with predicted spikelets number(top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

The same patterns are repeated in CQ_2017, presented in Fig. 5.12 where E2, which refers to Transfer Learning, yields the highest background errors with MSE of 5808.9 and MAE of 5741.92, while it also yields the lowest spike region errors with 135.72 for MSE and 91.05 for MAE. For the rest of experiments, the spike region based errors are on the same range, showing little effect for the different models tried.

In terms of combined error measures from Table 5.10, although initialising the model with ACID weights has produced the best performance within the spike region, training the model from scratch (E1) improved the overall performance with combined errors of 1165.53 and 1072.40 for MSE and MAE subsequently because of the reduced background errors. In terms of SMAPE, there is no significant difference among all experiments. Nonetheless, Transfer Learning has lead to the lowest relative percentage error of 97.56 due to the improvement in spike region spikelet estimation.

Optimal Set

For the Optimal set, there is a very significant improvement of model performance

compared to Original and Prediction because of the full elimination of the noisy background objects which make these images challenging. For CQ_2016 reported in Table 5.9, it is clear that using augmentation has improved the performance which led to the lowest error for MSE, MAE with a value of 70.68 and 69.13 respectively. In terms of SMAPE, it resulted to the lowest value of 15.67. In the case of CQ_2017 reported in Table 5.10 Transfer Learning gives the best results, followed by freezing the first block. This could be because with 5 scenes only for this stage, additional data in the form of ACID images increases performance. For CQ_2017 augmentation does not produce any improved results.

Prediction Set

Moving to the Prediction set for CQ_2016, overall it is training from scratch that produces the best results for MSE and MAE according to Table 5.9 with values of 555.07 and 543.19 respectively. A close look to the errors on the spike region (blue area of the bars) on Figure 5.11 illustrates that errors on that area are very similar for all experimental setups but may be slightly lower with augmentation. On the other hand, there is a significant change in background errors (red part of the bars) across all experiments. Training from scratch (E1) leads to the least errors on background areas with 251.49 for MSE, and 249.32 for MAE.

For CQ_2017, Table 5.10 shows that augmentation with Transfer Learning produces the best results overall, with values of 386.92 for MSE and 327.46 for MAE. For SMAPE, augmentation with training from scratch produces the best results with a value of 80.10. Figure 5.12 shows reduced errors with in the spike area when using data augmentation while training from scratch (123.64 for MSE and 85.15 for MAE) and also with Transfer Learning (155.99 for MSE and 102.73 for MAE). However augmentation leads to higher background errors than training from scratch. Lowest background errors are obtained while training from scratch with values of 182.99 for MSE and 181.50 for MAE.

Result comparison across all sets

In summary, booting is a difficult stage to analyse because the background becomes

very prominent given the lack of spikelets in images. Therefore best results are obtained when the background is removed completely (Optimal set) as that reduces background errors. This is particularly the case with Transfer Learning as that may help with spikelet identification given additional images from ACID. For the Original set, it is loading ACID weights and freezing the first block for CQ_2016 and training from scratch CQ_2017 that give best results as that reduces background errors while maintaining spike errors constant. For the Prediction set it is training from scratch for CQ_2016 or with augmentation and Transfer Learning in CQ_2017 that gives the best results. Differences could be due to different number of images available in each set.

5.4.4.2 G51-59: Heading

The Heading (ear emergence) is the sixth growth stage according to to the Zadok's growth decimal scale [141]. The Heading stage is considered a key development stage [85] which begins when the ear starts to emerge gradually from the flag leaf sheath until fully emerged [82]. Two key sub stages are important to recognise: half Heading, which is when 50% of the spike has emerged (G55), and full Heading when the full spike has fully emerged (G59). Figure 5.8 (c) and (d) showed two examples of spike emergence in Heading stage.

There are seven wheat images in the Heading growth stage in CQ_2016 with an average of 1464.40 spikelets as shown in Figure 3.7. For CQ_2017, there are also seven Heading images, which show more crowded spikelet scenes averaging 2970.12 spikelets per image as shown in Figure 3.8.

The analysis of Heading growth stage for CQ_2016 and CQ_2017 is reported in Table 5.11 and Table 5.12 respectively for three sets: Original, Optimal and Prediction. In addition, Figures 5.15 and 5.16 provide the quantitative errors for spike and background regions separately. Figures 5.17 and 5.18 show visualisation of the spikelet density maps for the Heading stage of 2016/2017 image series.

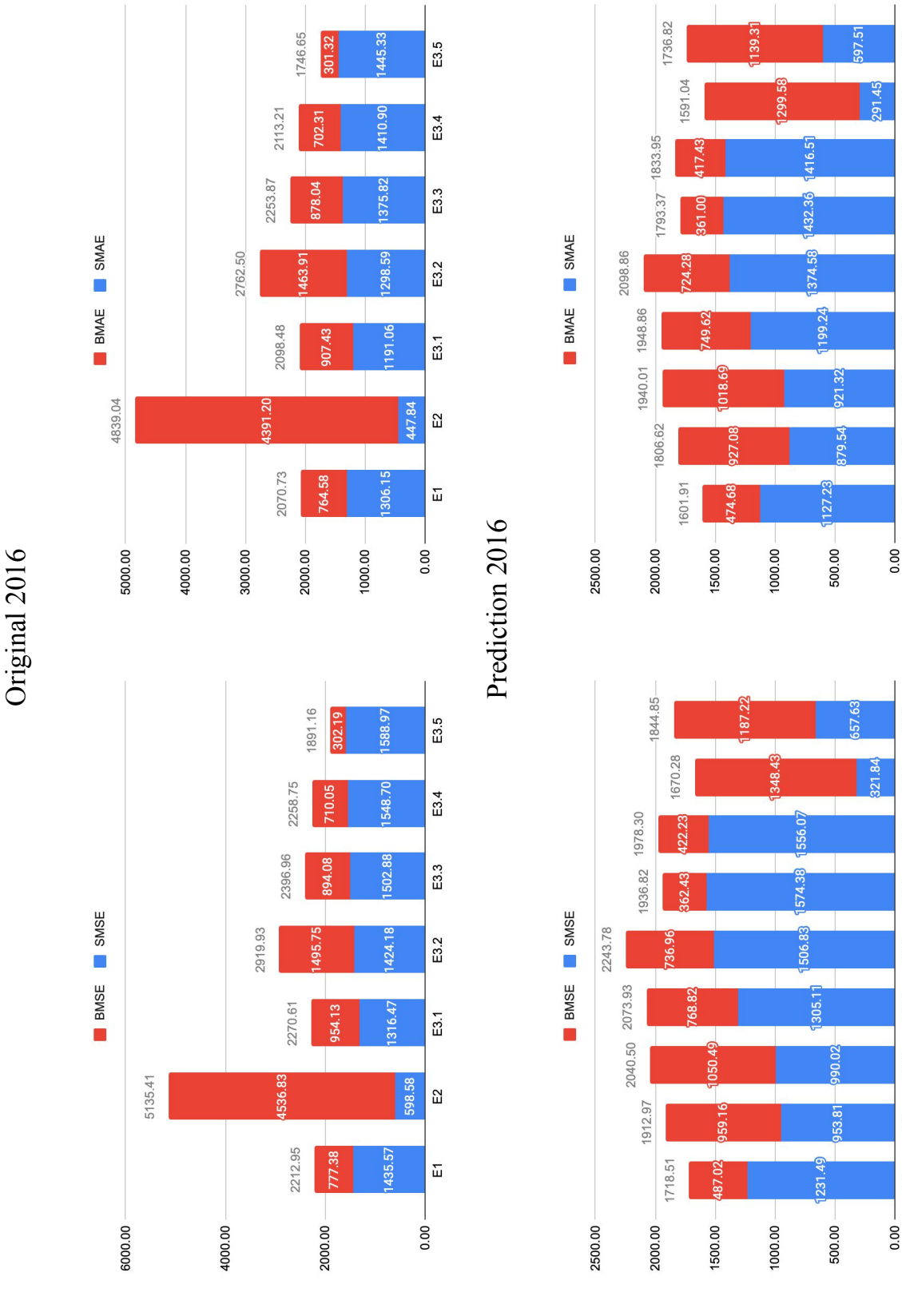

Figure 5.15: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Heading** growth stage-CQ_2016 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

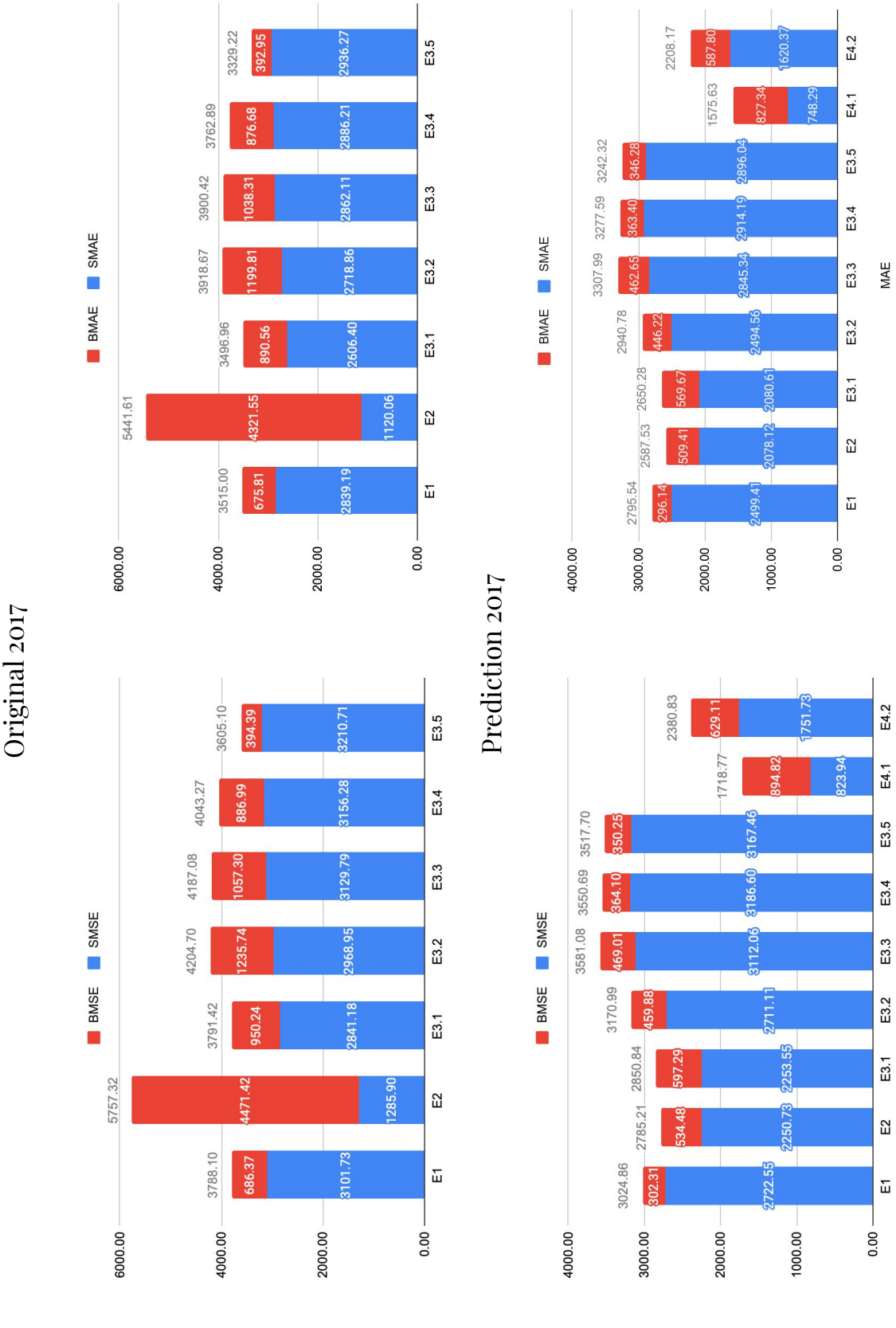

Figure 5.16: Heading Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Heading** growth stage-CQ_2017 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

Table 5.11: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for five experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on Heading growth stage of CQ_2016 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	**2212.95**	**2070.73**	88.35	812.22	717.45	31.88	1718.51	1601.91	72.91
Loading ACID Weights	5135.41	4839.04	**70.46**	308.93	255.21	10.606	1912.97	1806.62	64.38
Freezing Conv1 Blocks	2270.61	2098.48	82.67	307.74	254.84	10.608	2040.50	1940.01	67.59
Augmentation-Scratch	-	-	-	822.72	655.28	16.02	**1670.28**	**1591.04**	**46.09**
Augmentation-ACID	-	-	-	**302.57**	**214.11**	**6.70**	1844.85	1736.82	53.98

Table 5.12: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Heading** growth stage of CQ_2017 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	3788.10	3515.00	93.83	2061.41	1827.91	42.21	3024.86	2795.54	77.00
Loading ACID Weights	5757.32	5441.61	**64.95**	1245.19	1063.64	19.54	2785.21	2587.53	62.90
Freezing Conv1 Blocks	**3791.42**	**3496.96**	85.38	1256.00	1073.19	18.91	2850.84	2650.28	63.25
Augmentation-Scratch	-	-	-	892.80	830.18	13.76	**1718.77**	**1575.63**	**33.39**
Augmentation-ACID	-	-	-	**686.52**	**551.94**	**8.84**	2380.83	2208.17	49.61

Original Set

First, analysing the results for the Original set for CQ_2016, best performance for MSE and MAE is obtained by training the model from scratch. For CQ_2017, using ACID weights and freezing the first Block has led to best results. On the other hand for both CQ_2016 and CQ_2017 the lowest SMAPE is 70.46 and 64.95 respectively obtained by E2, Transfer Learning, as that takes into account the balance of errors in the background and spike areas. It is evident from Figures 5.15 and 5.16 that the second experimental setup, again for this stage as we saw with booting, results in over counting within the background region producing 4536.83 (MSE) and 4391.20 (MAE) within this region in CQ_2016; 2505.73 (MSE) and 2375.62 MAE in CQ_2017. On the other hand, the same experimental set up leads to the best performance when counting spikelets within the spike region with an MSE of 598.58 and MAE of 447.84 in CQ_2016; 1285.90 (MSE) and 1120.06 (MAE) in CQ_2017. Again this is similar to what we saw in the booting stage.

Optimal Set

For the Optimal set, as shown in Tables 5.11 and 5.12, it is clear that utilising

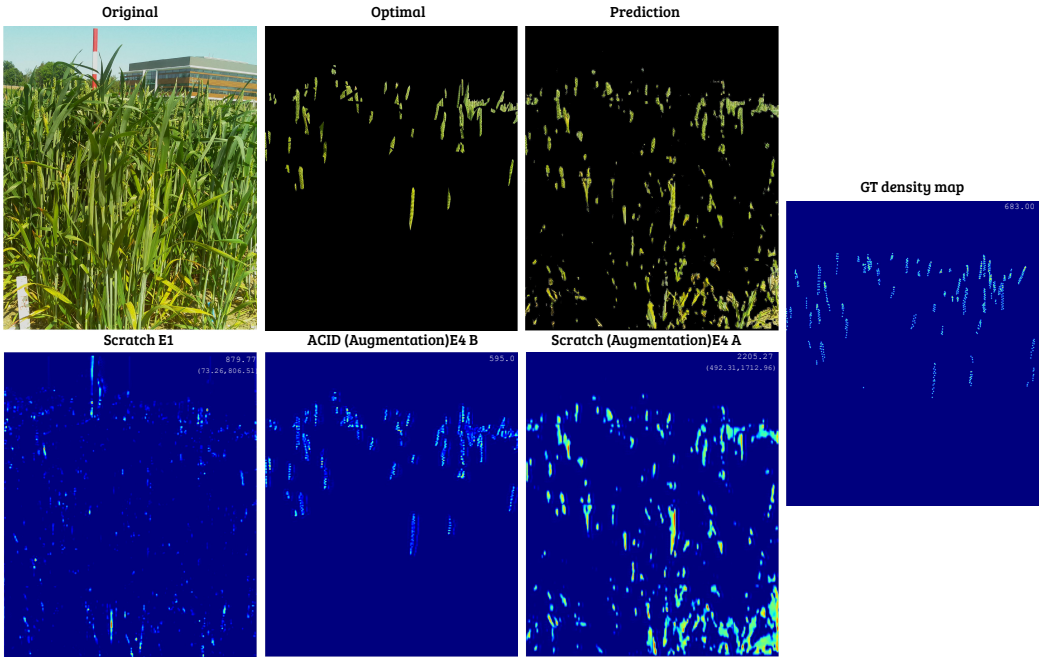

Figure 5.17: Visualisation of the spikelets density maps for the **Heading** stage of 2016 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction image while the second row shows the best resulting density map with predicted spikelets number(top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

both augmentation and ACID Transfer Learning produces the best results in this growth stage with MSE of 302.57 and MAE of 214.11 in CQ_2016; MSE of 686.52, MAE of 551.94 in CQ_2017. In addition, it leads to the lowest SMAPE of 6.70% in CQ_2016 and 8.84 in CQ_2017.

Prediction Set

For the Prediction set, overall it is augmentation used with training from scratch that produces the best models in CQ_2016 and CQ_2017. In the majority of experiments, according to Figures 5.11 and 5.12, the spike region errors are high. In contrast, augmentation has lead to better performance of the model in spike areas both in combination with training from scratch and Transfer Learning. Training from scratch with augmentation leads to the lowest MSE and MAE within spike regions for all experiments at 321.84 MSE and 291.45 MAE for CQ_2016; 823.94 MSE and 748.29 MAE for CQ_2017.

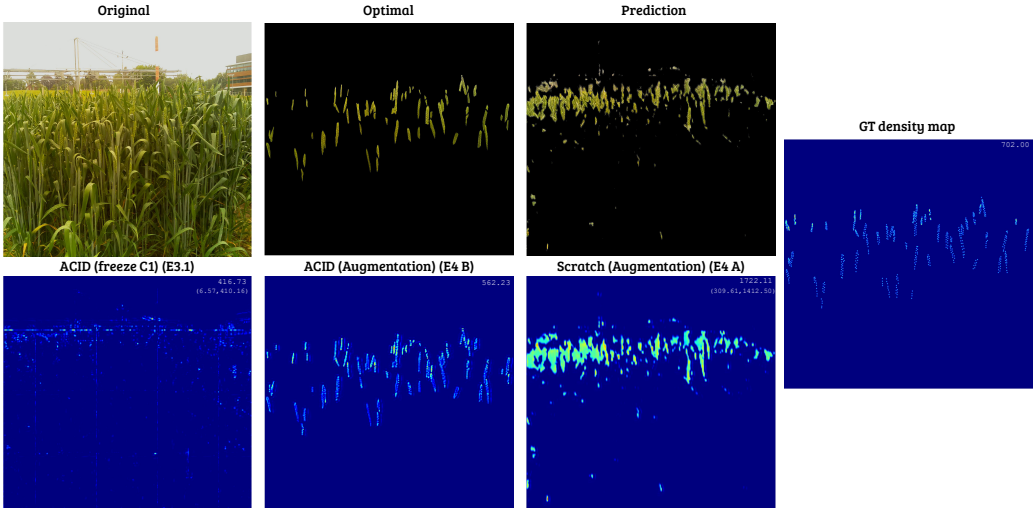

Figure 5.18: Visualisation of the spikelets density estimation maps for the **Heading** stage of 2017 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction image while the second row shows the best resulting density map with predicted spikelets number(top right. For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

Result comparison across all sets

In summary, for the Heading stage which has more spikelets to count in the images, best results by far are obtained by removing the background (Optimal set) once again. For the Original images, using Transfer Learning while freezing the first the block (E3.1) gives good results for CQ_2017, however, training from scratch (E1) gives good results for CQ_2016. For the Prediction set, data augmentation with training from scratch is the best option.

5.4.4.3 G61-69: Flowering (anthesis)

The Flowering is the seventh growth stage according to to the Zadok's growth decimal scale [141]. The Flowering stage is considered an key development stage [85] which begins after the Heading stage is complete and can be summarised according to Pask et al. [85] as: "when the 50% of spikes have extruded at least one anther and [...] mid-anthesis (mid Flowering) is recorded when 50% of spikes have extruded 50% of their anthers".

For this growth stage we have more representatives with 12 wheat images in CQ_2016 with an average of 2244.28 spikelets as shown in Figure 3.7. For CQ_2017, there are 13 Flowering images, which are more crowded, with spikelets counts averaging 3371.07 per image as shown in Figure 3.8.

Results for CQ_2016 are presented in Table 5.13 and for CQ_2017 in Table 5.14. Again we report each time for three sets: Original, Optimal and Prediction. Breakdown of errors for spikelet and background areas are presented in Figures 5.19 and 5.20. Figures 5.21 and 5.22 show visualisation of the spikelet density maps for the Flowering stage of 2016/2017 image series.

Table 5.13: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Flowering** growth stage of CQ_2016 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	2753.53	2709.59	88.20	1098.63	1050.07	30.20	1893.45	1846.88	57.43
Loading ACID Weights	3350.92	3066.16	**47.06**	406.91	357.91	8.11	1656.17	1546.81	38.85
Freezing Conv1 Blocks	**2523.45**	**2416.61**	85.05	**365.72**	**329.91**	**7.36**	1571.04	1466.18	34.20
Augmentation-Scratch	-	-	-	2561.33	2520.12	36.06	1972.68	1826.60	28.79
Augmentation-ACID	-	-	-	626.46	497.70	10.17	**1266.11**	**1133.29**	**24.44**

Table 5.14: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Flowering** growth stage of CQ_2017 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	3971.32	3920.14	92.19	2061.11	2016.67	42.58	2960.75	2915.79	67.57
Loading ACID Weights	5128.09	4679.23	**50.28**	831.05	713.41	11.63	2319.48	2230.08	41.91
Freezing Conv1 Blocks	**3723.46**	**3604.25**	83.77	**599.57**	**518.54**	**7.91**	2140.10	2019.60	35.52
Augmentation-Scratch	-	-	-	3250.78	3171.93	32.29	1769.35	1555.75	**18.70**
Augmentation-ACID	-	-	-	704.47	613.08	8.76	**1515.68**	**1306.21**	21.08

Original Set

For the Original set and for CQ_2016 it is clear from Figure 5.19, similar to Booting and Heading growth stage, that the second experimental setup, Transfer Learning, has resulted in over counting within the background region. On the other hand, for the spike region, the same experimental set up leads to the best performance with an MSE of 845.19 and MAE of 690.54. In terms of combined error measures from Table 5.13, we can see that for MSE the lowest errors resulted from freezing

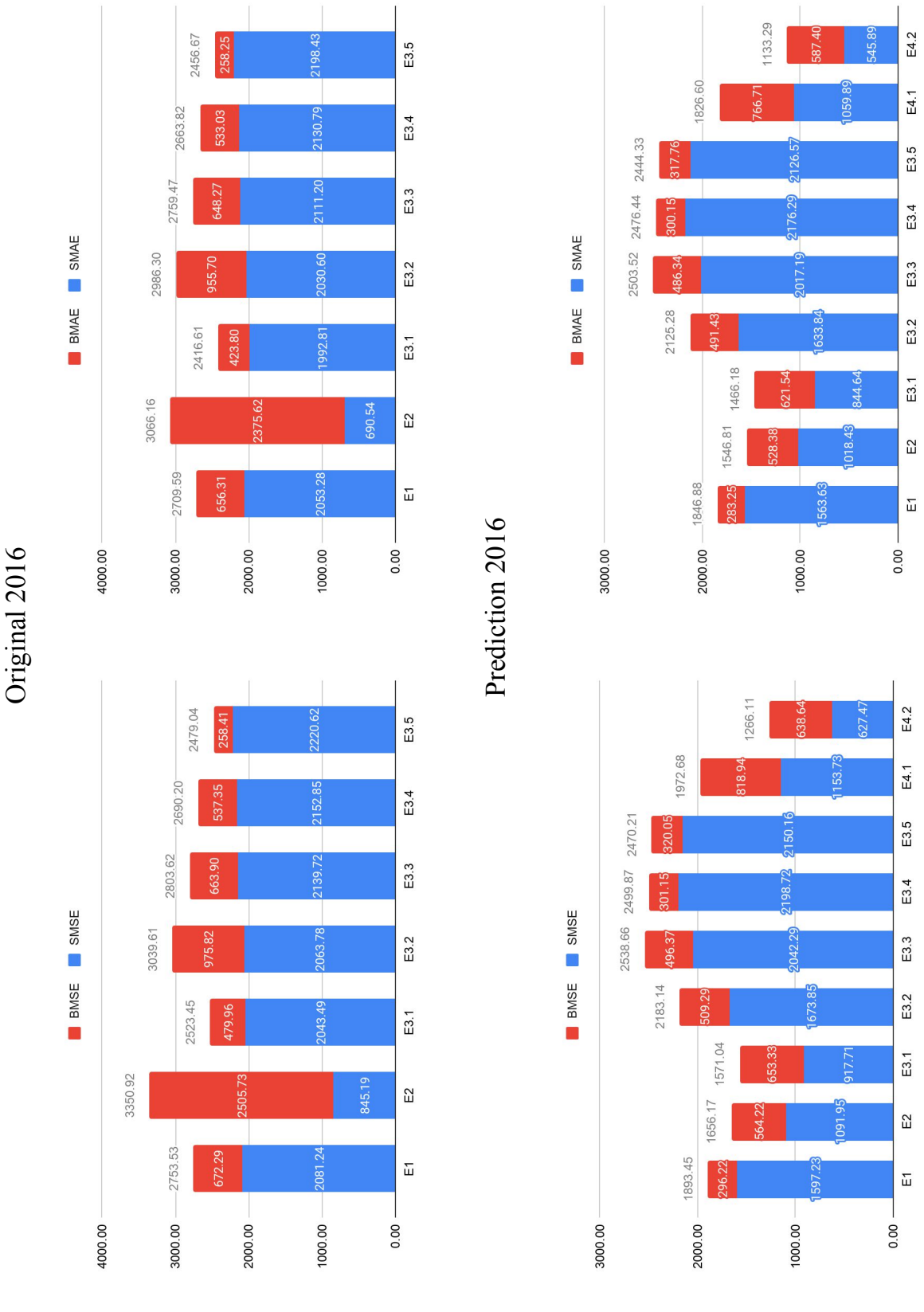

Figure 5.19: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Flowering** growth stage-CQ_2016 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

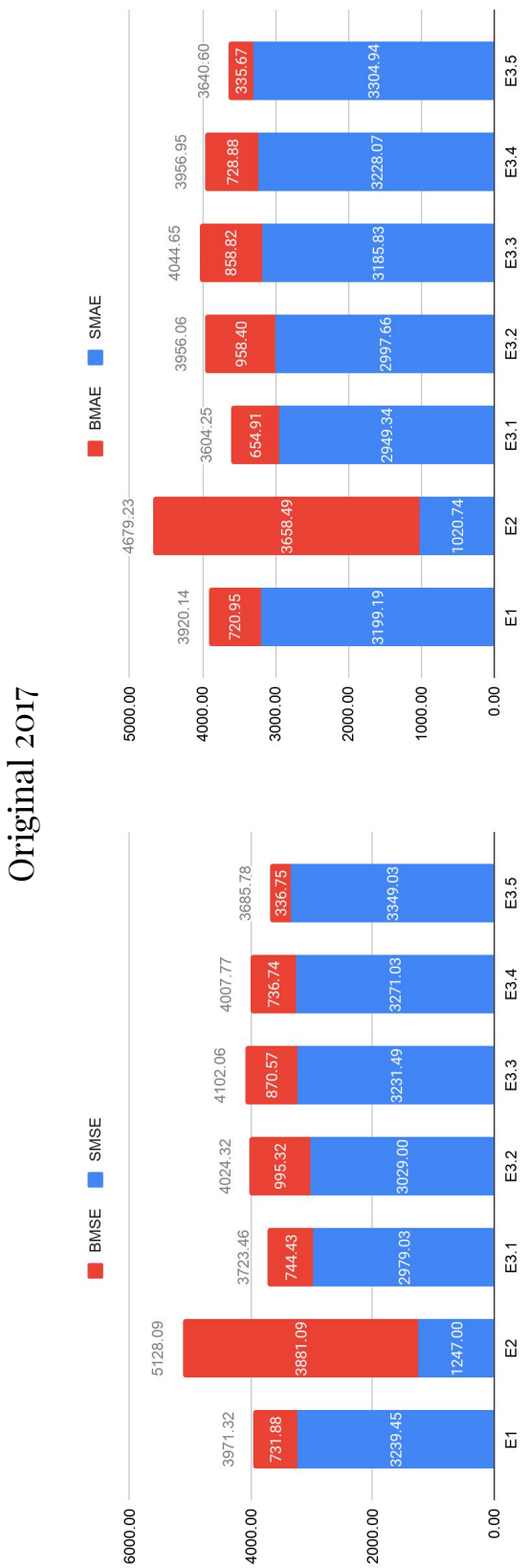

Figure 5.20: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Flowering** growth stage-CQ_2017 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights, E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

Conv1 block with value of 2523.45. For MAE, a value of 2416.61 is obtained while freezing only Conv1 block. The best SMAPE value of 47.06 is obtained by loading ACID weights as that lowers spike region errors and accurately counts spikelets within this region. The background errors when freezing conv1 block gave MSE value of 479.96 and MAE of 423.80 while the spike region MSE and MAE values are 2043.49 and 1992.81 respectively which are the second lowest.

The same insights apply for the results of Flowering stage for CQ_2017. Freezing only Conv1 as Table 5.14 shows results in combined MSE and MAE of 3723.46 and 3604.24 respectively. Also, Figure 5.20 shows the error breakdown of freezing conv1 block with MSE and MAE of 744.43 and 654.91 for background; for spike regions the MSE is 2979.03 and MAE is 2949.34.

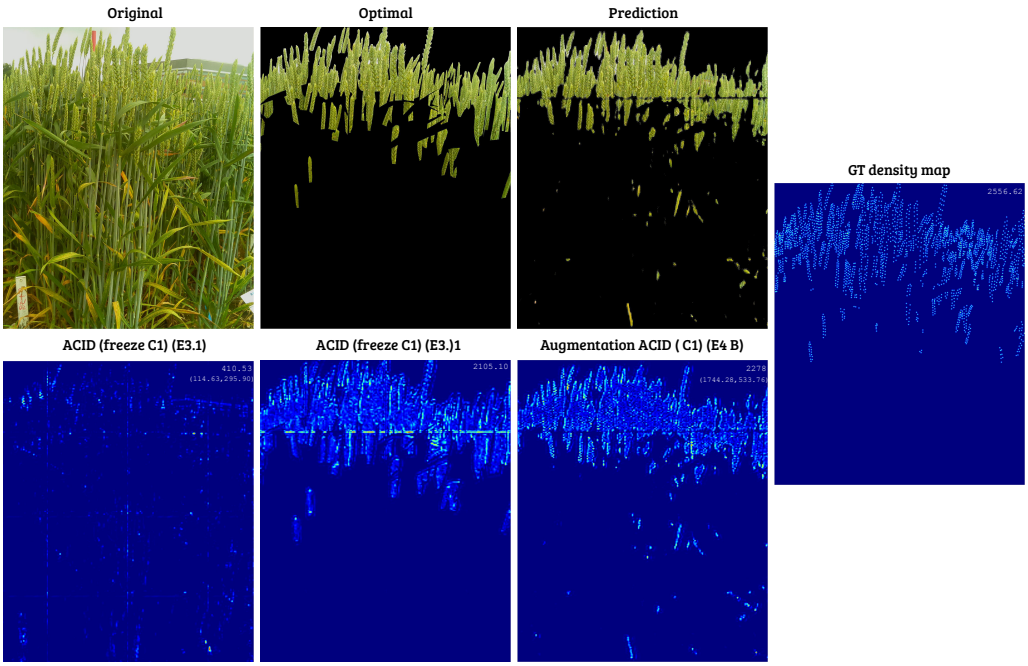

Figure 5.21: Visualisation of the spikelets density maps for the **Flowering** stage of 2016 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction set while the second row shows the best resulted density map with predicted spikelets number(top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

Optimal Set

For the Optimal set, as shown in Table 5.13 and Table 5.14 freezing Conv1 block

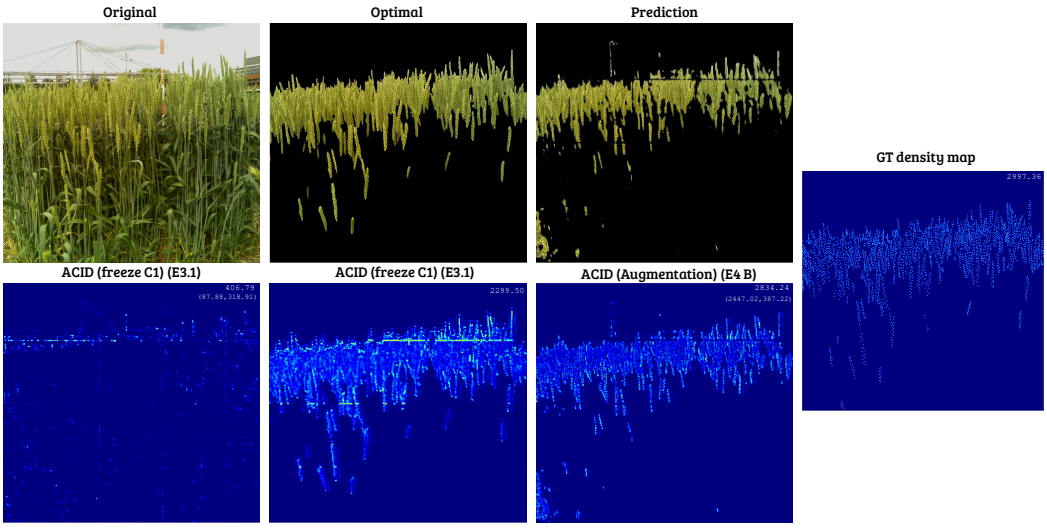

Figure 5.22: Visualisation of the spikelets density maps for the **Flowering** stage of 2017 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction set while the second row shows the best resulted density map with predicted spikelets number (top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

is the best overall performer for both years which has significantly improved the model performance with MSE of 365.7 and MAE of 329.91 in 2016. In addition, it led to the lowest SMAPE of 7.36 %. On the other hand, only initialising the model with ACID weights has led to the second best performance. Similarly in 2017, freezing Conv1 block has improved the model performance with MSE of 599.57 and MAE of 518.54, and SMAPE of 7.91%.

Prediction Set

Moving to the Prediction set, according to Table 5.13, the best performance overall is obtained by using augmentation in combination with Transfer Learning using ACID weights with values of 1266.11 and 1133.29 for MSE and MAE in 2016. Also, with a relative error percentage of 24.44% for CQ_2016. In CQ_2017 that also leads to the lowest values of MSE and MAE with values of 1515.68 and 1306.21 respectively. SMAPE is the best, however, when augmentation is used with training from scratch.

Result comparison across all sets

In summary, for the Flowering growth stage we have more images and high number of spikelets to count. Again, removing the background altogether, as we do for the Optimal set, results in the best errors. For the Optimal set it is using ACID images in combination with freezing the first Block that results in the best performance in both years. For the Original set, freezing only the first block gives good results. For the Prediction set, it is augmentation with ACID weights that gives the best results. So augmentation is helpful for both the Heading and the Flowering stage, but for the Flowering stage the addition of ACID weights through Transfer Learning also benefits the models. The results of Optimal set are better than the Prediction set results combined results (Tables 5.13 and 5.14). However, when comparing to the spike region results, we can see it shrink the gap between the two set results (Figures 5.19 and 5.20).

5.4.4.4 GS71-73: Grain filling

The Grain filling is the eighth growth stage according to to the Zadok's growth decimal scale [141]. The Grain filling stage is considered a key development stage [85] which begins after the Flowering stage is complete and is related to the developments of grains [82]. Our dataset for all sequences only contains sub stages from G71 to G73 which are stages when most of Grains contain watery fluids [85]. There are 9 wheat images in the Grain filling growth stage in CQ_2016 with an average of 2560.75 spikelets as shown in Figure 3.7. For CQ_2017, there are 5 Grain filling images with spikelets counts averaging 3454.62 as shown in Figure 3.8.

Results for this stage are reported in Table 5.15 for CQ_2016 and Table 5.16 for CQ_2017 for three sets: Original, Optimal and Prediction. In addition, Figures 5.23 and 5.24 provide the quantitative errors from the spike and background regions separately for each year. Also, Figures 5.25 and 5.26 show visualisation of the spikelets density maps for the Grain filling stage of 2016/2017 image series.

Original Set

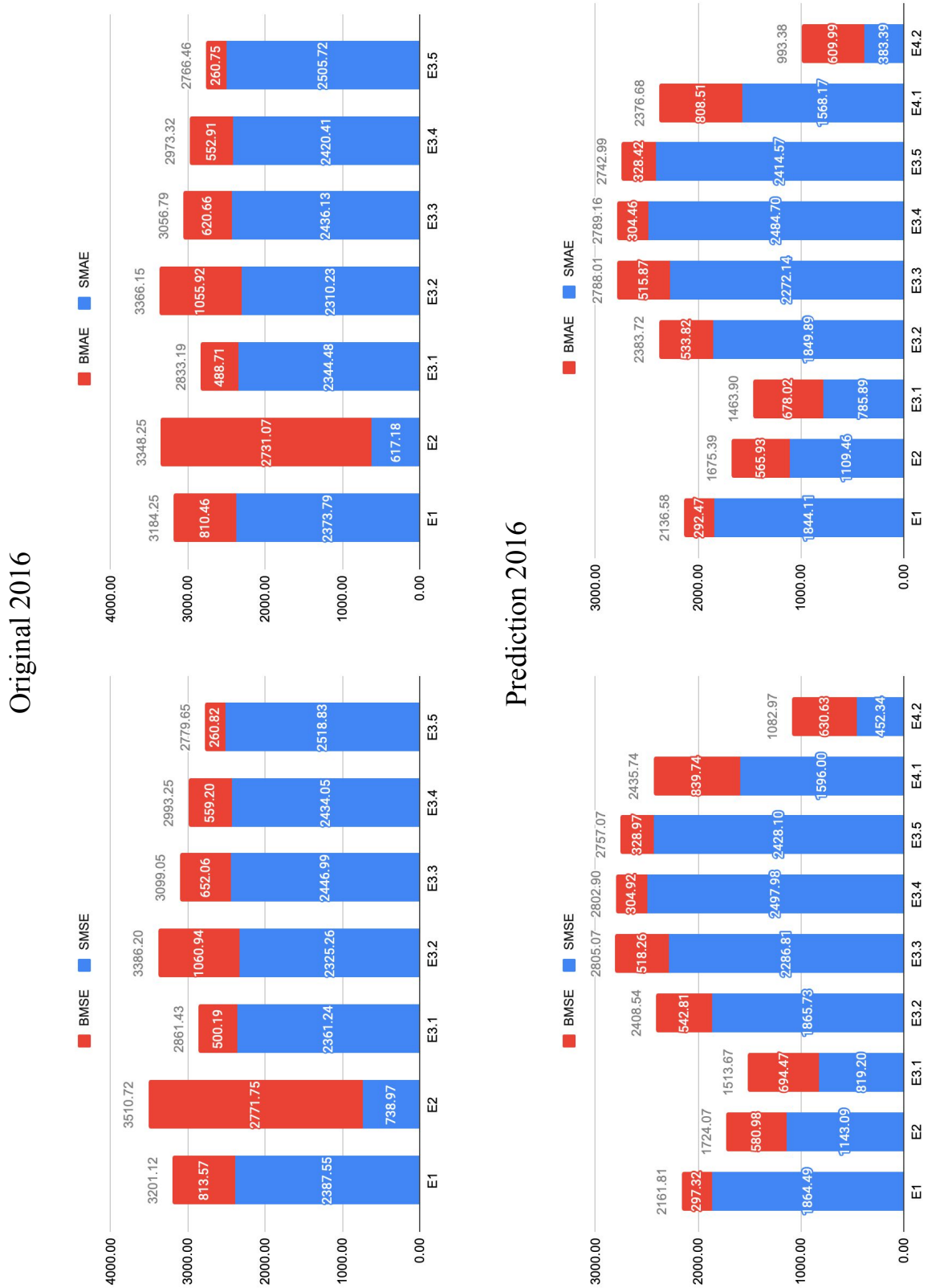

Figure 5.23: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Grain filling** growth stage-CQ_2016 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights, E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

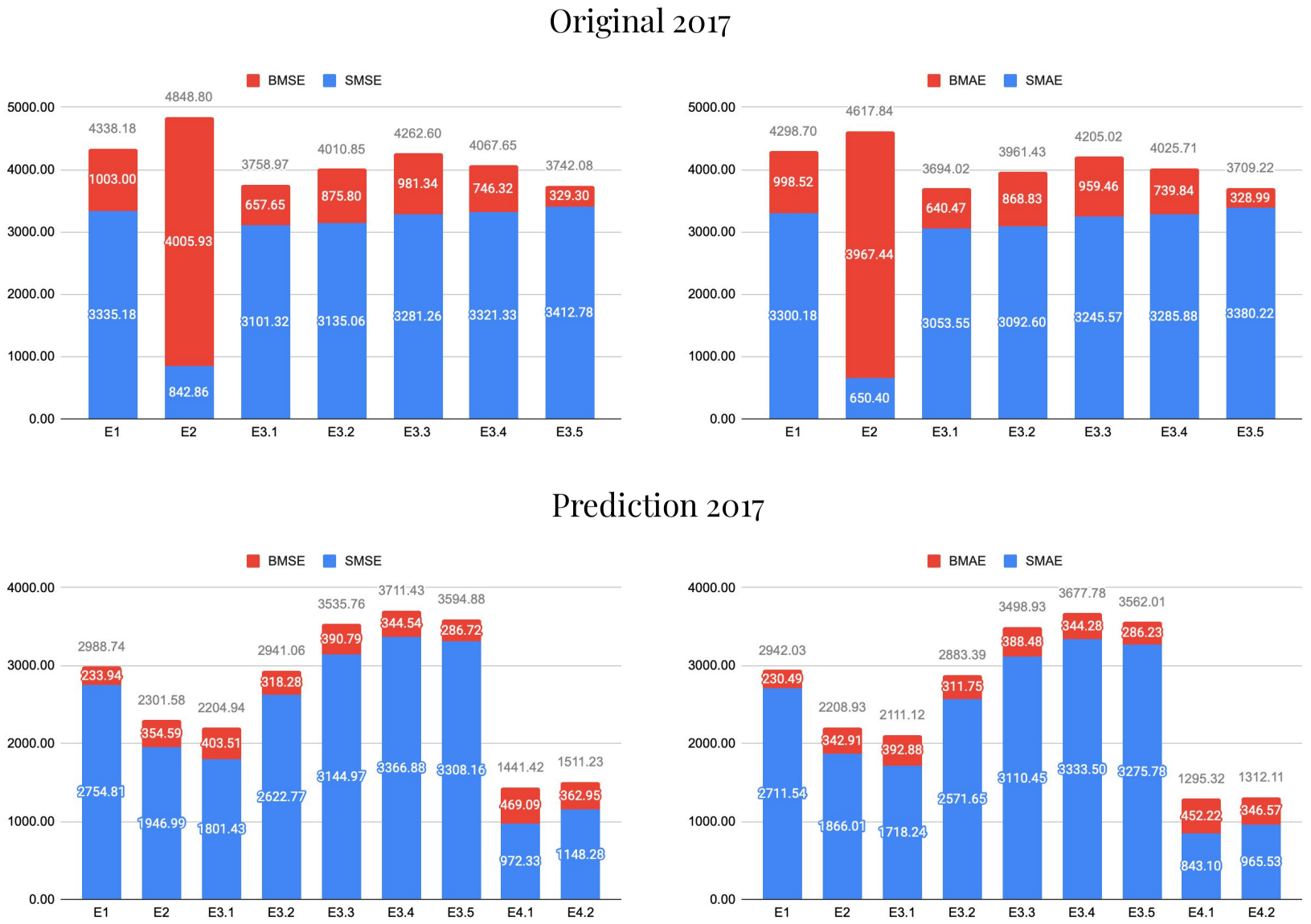

Figure 5.24: Background and spike region errors breakdown of four experimental setups for Original and pre-segmented (predicted) sets of **Grain filling** of CQ_2017 sequence where E1 refers to training From Scratch, E2 refers to loading ACID Weights , E3.1 refers to freezing Conv1 Block, E3.2 refers to freezing Conv2 Block, E3.3 refers to freezing Conv3 Block, E3.4 refers to freezing Conv4 Block, E3.5 refers to freezing Conv5 Block, E4.1 refers to augmentation-Scratch and E4.2 refers to augmentation-ACID.

Table 5.15: The Mean Squared Error, Mean Absolute Error, and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Grain filling** growth stage of CQ_2016 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	3201.12	3184.25	89.60	1413.25	1384.76	36.79	2161.81	2136.58	59.69
Loading ACID Weights	3510.72	3348.25	**45.06**	341.95	282.95	5.64	1724.07	1675.39	36.52
Freezing Conv1 Blocks	**2861.43**	**2833.19**	87.08	**276.80**	**232.72**	**4.56**	1513.67	1463.90	29.09
Augmentation-Scratch	-	-	-	2980.53	2972.47	36.86	2435.74	2376.68	31.57
Augmentation-ACID	-	-	-	742.85	668.14	11.64	**1082.97**	**993.38**	**18.32**

Table 5.16: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets for four experimental setups (displayed as rows) of the Original, pre-segmented by ground truth (Optimal) and pre-segmented by model (predicted) images (displayed in columns) on **Grain filling** growth stage of CQ_2017 sequence

	Original			Optimal			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE
Training From Scratch	4338.18	4298.70	93.22	2036.07	1979.84	39.74	2988.74	2942.03	66.16
Loading ACID Weights	4848.80	4617.84	**43.56**	802.25	593.29	9.07	2301.58	2208.93	40.87
Freezing Conv1 Blocks	**3758.97**	**3694.02**	82.35	**502.07**	**321.02**	**4.49**	2204.94	2111.12	37.69
Augmentation-Scratch	-	-	-	3712.86	3685.36	35.04	**1441.42**	**1295.32**	**15.93**
Augmentation-ACID	-	-	-	841.17	811.88	10.89	1511.23	1312.11	20.89.

First, for the Original set, best results for CQ_2016, according to Table 5.15, are obtained by freezing Conv1 block with a MSE of 2861.43 and MAE of 2833.19. For CQ_2017 best results are obtained also by freezing Conv1 block with a MSE of 3758.97 and MAE with a value of 3694.02. The SMAPE measure is best for Transfer Learning (E2) with a value of 45.06 for CQ_2016 and 43.56 for CQ_2017.

Looking at the break down of errors in the corresponding Fig. 5.23 as before, it is clear that E2 leads to very high errors on the background and low errors on the spike area. Freezing Con1 Block background errors are 500.19 for MSE and 488.71 for MAE. The best overall SMAPE is obtained when using ACID weights, E2, as that leads to a reduction in spike area errors. The same observations can be made for CQ_2017 as reflected in Fig. 5.24 in terms of problems with E2 which maximises background errors, freezing Conv1 has led to balanced results of both background and spike regions errors. Again SMAPE is best for E2.

Optimal Set

The Optimal set is again the best performing, as shown in Table 5.15. Consistently with other stages, freezing Conv1 Block has significantly improved the model per-

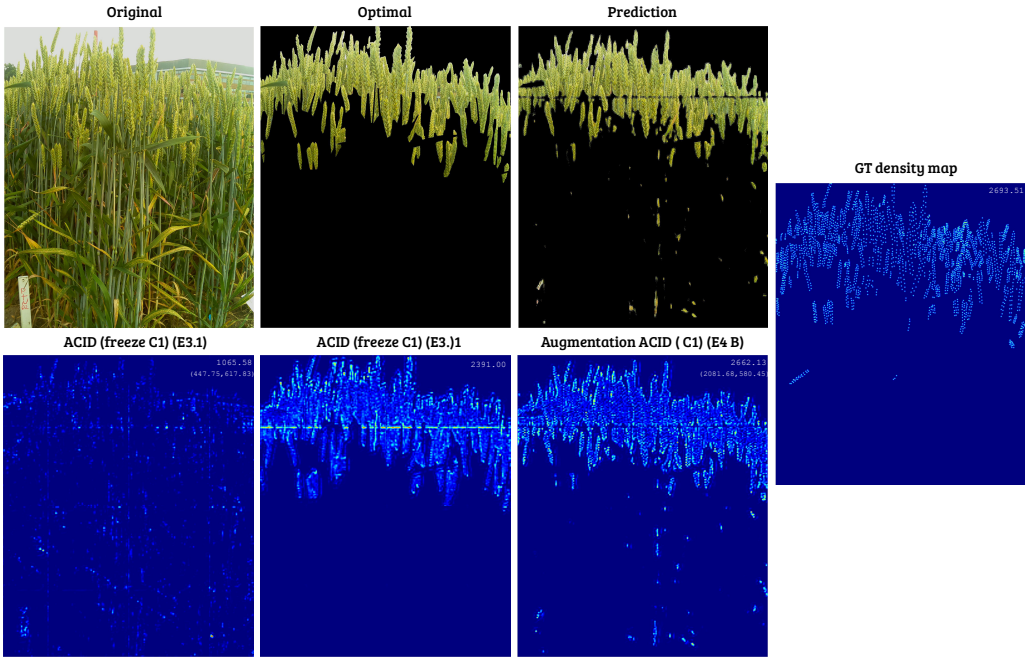

Figure 5.25: Visualisation of the spikelets density maps for the **Grain filling** stage of 2016 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction set while the second row shows the best resulting density map with predicted spikelets number(top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

formance with MSE of 276.80, MAE of 232.72 and SMAPE OF 4.56 for CQ_2016; MSE of 502.07, MAE of 321.02 and SMAPE of 4.49 for CQ_2017 .

Prediction Set

For the Prediction set, best results are obtained by using augmentation while training with ACID weights in CQ_2016 and augmentation while training from scratch in CQ_2017. Looking at the breakdown of errors in Figures 5.23 and 5.24 in the majority of experiments, the spike area errors are high particularly for training from scratch (E1). In contrast, for CQ_2016 augmentation and loading ACID weights at the same time has led to significant improvements in the performance of the model. For instance, the MSE and MAE within spike regions are 452.34 and 383.39 which are the lowest across all experiments. For CQ_2017, both combinations of augmentation produce good results though they are slightly better when training from scratch as that results in lower errors in the spike areas.

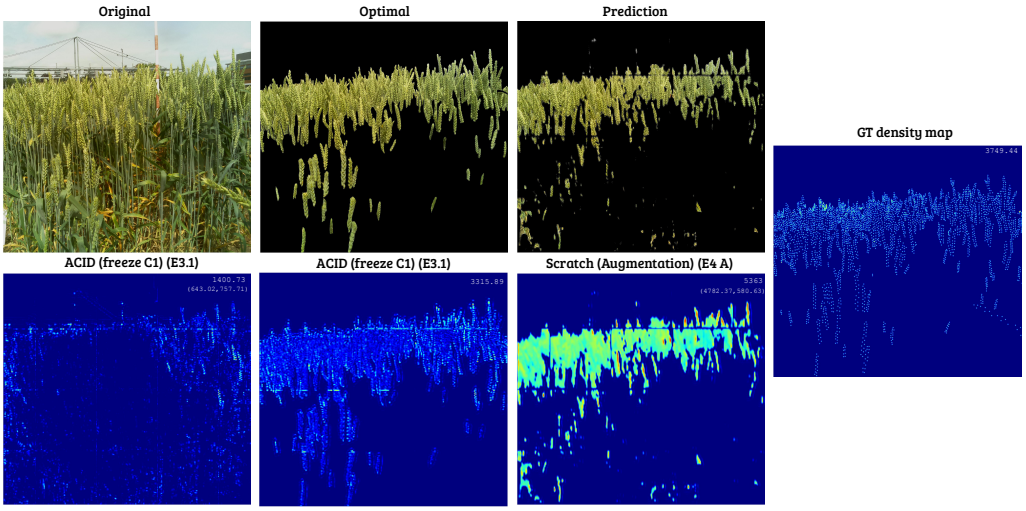

Figure 5.26: Visualisation of the spikelets density maps for the **Grain filling** stage of 2017 image series. The first row shows from left to right: (1) Original image, (2) Optimal image, (3) Prediction image while the second row shows the best resulting density map with predicted spikelets number(top right). For Original and Prediction images the counting is detailed as follows. Top is the total number and bottom is predicted spikelet number within spike region and predicted spikelet number within background when SpikeCount is tested on the corresponding image shown in the first row. The ground truth density map and number of spikelets is presented rightmost.

Result comparison across all sets

In summary for the Grain filling stage, Optimal is as always best as it removes background completely allowing the algorithm to concentrate on the spike area. In this scenario, using ACID weights while freezing Conv1-Block gives best results. For the Original set, freezing conv1 block has led to the best results in terms of combined errors, while errors on the spike area are given more prominence by SMAPE, so for this particular metric loading ACID weights alone gives best performance. For the Prediction set, it is some form of augmentation that gives best results.

5.4.4.5 Comparison across all growth stages

Table 5.17 provides a comparison of which experimental setup of SpikeCount has worked best for each quantitative metric across Original, Prediction and Optimal sets for each growth stage for CQ_2016 and CQ_2017.

For 2016 growing season and Optimal set, freezing the conv1 block after loading

Table 5.17: A comparison of best experimental setups of SpikeCount for each quantitative metric across Original, Prediction and Optimal sets for each growth stage for CQ_2016 and CQ_2017

2016									
	Original				Prediction				Optimal
Growth stage	Best Background Errors	Best Spike Errors	Best Combined	Best relative error %	Best Background Errors	Best Spike Errors	Best Combined	Best relative error %	Best Errors and relative error %
Booting	E3 C1	E2	E3 C1	E2	E1	E4 A	E1	E4 A	E4 A
Heading	E1	E2	E1	E2	E1	E4 A	E4 A	E4 A	E4 B
Flowering	E3 C1	E2	E3 C1	E2	E1	E4 B	E4 B	E4 B	E3 C1
Grain filling	E3 C1	E2	E3 C1	E2	E1	E4 A	E4 B	E4 A	E3 C1
2017									
	Original				Prediction				Optimal
Growth stage	Best Background Errors	Best Spike Errors	Best Combined	Best relative error %	Best Background Errors	Best Spike Errors	Best Combined	Best relative error %	Best Errors and relative error %
Booting	E1	E2	E1	E2	E1	E4 A	E4 B	E4 A	E2
Heading	E1	E2	E3 1	E2	E1	E4 A	E4 A	E4 A	E4 B
Flowering	E1	E2	E3 1	E2	E1	E4 B	E4 B	E4 A	E3 C1
Grain filling	E3 C1	E2	E3 1	E2	E1	E4 A	E4 A	E4 A	E3 C1

pretrained ACID parameters has resulted in the best results of both errors metrics and relative error percentage for Flowering and Grain filling growth stage while using augmentation after initialisation with ACID parameters has resulted in the best outcome for Heading growth stage. For Booting growth stage only using augmentation has lowered the errors metrics.

Observing Original set, for Booting, Flowering and Grain filling, freezing conv1 block has resulted in lowest background errors. When we combine both spike regions and background errors while initialising the model with pretrained ACID parameters it has minimised the spike regions errors and relative error percentage for all growth stages. Training from scratch has lowered the background and combined errors for the Heading growth stage.

Moving to the Prediction set, training from scratch has minimised the background errors for all growth stages and also minimised the combined results for only Booting growth stage. Employing augmentation has resulted in lowering spike regions errors for all growth stages except Flowering and also lowering combined errors for Heading and Grain filling stages. Also, it has lowered the relative error % percentage for all growth stages except Flowering. Utilising augmentation and transfer ACID learnt features has minimised the spike regions error, combined error, and relative error percentage for Flowering growth stage.

For 2017 growing season and Optimal set, freezing the conv1 block after loading pretrained ACID parameters has resulted in the best results of both errors metrics and relative error percentage for Flowering and Grain filling growth stage while using augmentation after initialisation with ACID parameters has resulted in the best outcome for Heading growth stage. Only initialising ACID parameters has improved the results for Booting stage.

For the Original set, for Booting, Heading and Flowering, training from scratch has resulted in the lowest background errors and also it has lowered booting combined errors. Freezing conv1 block has lowered the Grain filling background errors and

Heading, Flowering and Grain filling combined errors. Training by initialising the model with pretrained ACID parameters has minimised the spike regions errors and relative error percentage for all growth stages.

For the Prediction set, training from scratch has minimised the background errors for all growth stages. Employing augmentation has resulted in lowering spike regions errors for all growth stages except Flowering and also lowering combined errors for Heading and Grain filling stages. Also, it has lowered the relative error% percentage for all growth stages. Utilising augmentation and transfer ACID learnt features has minimised the spike regions error for only Flowering and has lowered combined error for booting and Flowering growth stages.

5.5 Summary

Counting spikelets from infield wheat crop images is a vital step in quantifying yield traits which can help in monitoring wheat crop growth through automated systems. This is a very challenging task given the variability, self-similarity, varying backgrounds, severe occlusion, density, and changes in illumination etc. associated with spikelets in real wheat images. In this Chapter, we introduced SpikeCount, a fully convolutional network to count spikelets using a density estimation approach. We have designed several experimental setups to evaluate the model performance such as investigating the effect of transferring learnt features from wheat images captured in controlled environment using Transfer Learning. In addition, we investigated which feature level has the most effect in improving the results. Finally, we investigated the importance of augmentation. We have also conducted the evaluation on three variation sets (Original (images with background), Optimal (images with predefined segmentation) and Prediction (the output prediction from the segmentation model discussed in Chapter 4))of two growing seasons 2016 and 2017.

From analysing the result, it is clear that isolating ROI, represented by evaluating

on Optimal set, has led to the best results compared to Prediction and Original for all experiments for both CQ_2016 and CQ_2017. The second best results are noted when evaluating on Prediction set. Not removing the background as in Original set has resulted in the worse results which indicate that isolating the ROI (i.e spike regions) play an important role in solving this problem in this context.

In terms of the best factor for Optimal, freezing the conv1 block has resulted in best results for CQ_2016 (and to some extent for CQ_2017) which indicate that the transferred lower features from the ACID parameters have important impact on helping to extract the same traits from images with the same domain but captured in an uncontrolled environment. Also, utilising augmentation and Transfer Learning has led to a close performance to the results of freezing conv1 block.

Moving to the results of Prediction set, it is clear that employing augmentation and ACID Transfer Learning has a positive impact for spike regions but for background errors, those are minimised by training from scratch for both growing seasons.

Freezing conv1 block has led to lowering combined and background errors for the Original sets of both CQ_2016 and CQ_2017. However, from Figure 5.4, we observed that freezing the model encoder with all its variations has led to severe under-counting. This indicates that freezing the encoder block(s) hinders the model learning when trained on images with full background. This could be due to difference between images in Original set and ACID which is captured in a background free environment. In terms of spike region errors for both growing seasons, initialising the model with ACID parameters has minimised them significantly.

Analysing the results based on growth stages of the Optimal set shows that for more mature wheat images freezing the first block in SpikeCount after loading ACID weights has led to the best performance for both 2016 and 2017 growing seasons. Regarding Booting growth stage in CQ_2017, initialising the model with ACID weights and continuing to train has improved the results of this stage. Finally,

transferring ACID knowledge and augmentation has improve the Heading growth stage results for both seasons and only employing augmentation has improved the counting results of the Booting stage in CQ_2016.

All in all, it is evident that both eliminating the background and transferring the knowledge learnt from the ACID datasets has a significant effect on counting spikelets from infield uncontrolled images. Also, employing augmentation shows advantage in specific scenarios. On the other hand, there is a need to improve the results in relation to the Original set and this could be tackled through a supervised MTL approach. This would apply both segmentation and density estimation simultaneously which will be discussed in the following Chapter (Chapter 6).

Chapter 6

Heterogeneous Multitask Infield Wheat Phenotyping

It is evident from the analysis of Chapter 5 results that isolating the ROI (i.e. spike regions in our case) completely, as it is done using the optimal set of the CropQuant (CQ) sequences, has produced the best results across all our experimental variations. This indicates the importance of isolating an ROI in solving the challenge of spikelet counting. This eliminates noisy background that could interfere with achieving accurate spikelet counting, especially in infield settings. In Chapter 5 we tested this principle in two ways:

1. The first was to provide a extreme case with no isolation of the spike region. We tested SpikeCount directly on images without removing any background, the Original set. We found that testing directly on the Original set lead to the worst results, as may be expected due to the noisy background.

2. In the second case we used a conventional pipeline by using a model, a SpikeSEG from Chapter 4, to produce a segmentation and isolate ROI. Then we applied the second model, SpikeCount from Chapter 5, to the output from the first model to extract the required traits (i.e. spikelets numbers in our case). We evaluated this as the Prediction set. This approach has led to the second best results.

To further improve performance for both tasks (spike segmentation and spikelet numbers estimation), we now propose the concept of multitask learning.

Multi-task Learning (MTL) is a branch of machine learning, where multiple related targets are trained simultaneously to solve complex goals [15]. Caruana [15] argues that when it comes to solving complex goals, relying on target modularity and solving each one independently can be ineffective. Also, MTL can help develop more accurate models for each task by leveraging useful information learnt from related tasks [147]. Pan and Yang [83] has considered MTL as a branch of inductive Transfer Learning where the goal is to learn a low-dimensional feature representation which is shared among different targets.

According to Zhang and Yang [147] there are two factors that determine MTL setting: first, tasks relatedness which explores how different tasks are related and, second, tasks learning type such as supervised tasks, unsupervised tasks, active tasks, reinforcement tasks and other learning task types. One popular task learning type is multi-task supervised learning (MTSL), and that is what we are using in this work.

There are three aspects of MTSL task relatedness: features, parameters and instances [147]. We will focus on feature-based MTSL which determines how different tasks share a feature representation from the Original feature set.

Feature-based MTSL can also be categorised further according to Zhang and Yang [147] into: feature transformation, feature selection and deep learning approach. Feature transformation MTL involves learning a linear or nonlinear transformation from original features to the shared feature representation among the targets using multi-layer feed-forward neural network. Feature selection MTL is concerned with selecting the shared feature representation between different tasks from the original feature representation. Deep learning MTL is similar to feature representation using multi-layer feed-forward neural network but the shared feature representation is learned using more complex models such as convolutional neural networks and

recurrent neural networks that may consist of hundreds of hidden units.

Yang et al. [137] has divided MTL into two categories: homogeneous and heterogeneous in terms of the type of target. A homogeneous MTL model consists of similar targets (outputs) either discrete to solve classification problems or continuous to solve regression problems. In contrast, a heterogeneous MTL model consists of different types of output and can tackle both types, regression and classification, at the same time. Furthermore, Zhang and Yang [146] have extended the differences between heterogeneous and homogeneous MTL to include the type of target learning used, whether it is supervised/unsupervised learning and other types of learning.

In the context of object counting, Zhang et al. [143] developed a homogeneous MTL algorithm with multiple learning objectives which are switchable. Their objectives are crowd density and crowd counting and they can help each other to obtain better estimates. An example of heterogeneous object counting MTL is the method of Arteta et al. [6] to count penguins. They introduced a multitask fully convolutional Neural Net architecture that primarily learns a segmentation mask to help the object density map regression. They then generate the labels for the three targets from crowd-sourced dot-annotations. More specifically, the model is trained first to learn to predict a segmentation map and then fine tuned to regress a density and uncertainty map. Pound et al. [90] developed a homogeneous MTL deep learning model to count and localise wheat spikes and spikelets, achieving an accuracy of 95.91% and 99.66% respectively using Non-Maximal Suppression (NMS). They tested the model on The Annotated Crop Image Dataset (ACID) captured in a controlled environment inside a glasshouse.

In this Chapter, we propose SpikeMulti, a heterogeneous multi-task fully convolutional model that is trained simultaneously to learn two tasks: (1) pixel-wise classification for spike segmentation and (2) pixel-wise regression for spikelet density estimation used to infer spikelets counts. We hope this approach will contribute to learning more generalised features that could help both tasks: segment spikes

and count spikelets at the same time. We will study whether this approach could improve spikelet counting for the Original set and whether the segmentation quality can be improved for CQ_16/17. Section 6.1 summarises the datasets we used, the multi-task architecture of SpikeMulti, the experimental setup, model optimisation and training procedure details. Section 6.3 describes the performance results of testing SpikeMulti and the their interpretation for each experimental setup. Also, we discuss results of testing SpikeMulti for each growth stage. Finally, Section 6.4 presents our conclusions from these experiments.

6.1 Methods

In this section, first we lay out the definition of MTSL for SpikeMulti mathematically. Then, we explain the SpikeMulti architecture, the specification of model training including the cost function, the training hyperparameters, and the protocol of splitting the sequences into training, validation and testing sets. In addition, we explain in detail each experimental setup and its aim.

6.1.1 Multi-task supervised learning (MTSL)

We follow Zhang and Yang's [147] definition of MTSL. For our SpikeMulti the m supervised learning tasks are $m = 2$, so we have $T = \{T_1, T_2\}$. The first learning task T_1 represents spike segmentation which is a pixel level classification and is associated with a training dataset $D_1 = \{(X_j, y_j^1)\}_{j=1}^n$, where (X_j) is a wheat image represented as $\{M \times N\}$ RGB pixels and y_j^1 is the segmentation label of (X_j) which is a pixel wise encoding $y_j^1 \in \{0, 1\}$ where 0 represents non-spike region and 1 is a spike region.

The second learning task T_2 represents spikelets density estimation pixel level regression and is associated with a training dataset $D_2 = \{(X_j, y_j^2)\}_{j=1}^n$, where (X_j) is a wheat image represented as $\{M \times N\}$ RGB pixels and y_j^2 is the continuous

density label of (X_j) which is a pixel wise real scalar which is generated from dot annotation.

6.1.2 SpikeMulti Architecture

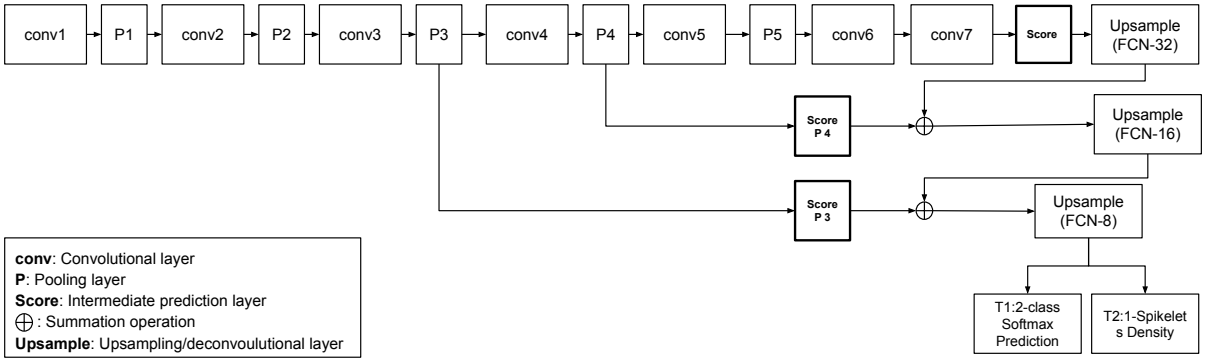

Figure 6.1: SpikeMulti Architecture

Our SpikeMulti, Figure 6.1, uses the same architecture as the spike segmentation and spikelet counting we used and explained in detail in Chapters 4 and 5. The only addition is two output layers to represents both supervised learning tasks: the pixel level softmax layer responsible for prediction of spike segmentation and pixel level regression to predict a density estimation for each pixel.

6.1.2.1 Global Cost Function

In order to determine the global cost function to optimise any deep learning MTSL L, we need to determine the partial cost functions L_i for each learning task T_i. In addition, we need to determine the weight loss coefficient λ_i for each learning task to distinguish the importance of tasks when training. Therefore, The global cost function is the weighted summation of all partial learning task cost functions (Equation 6.1):

$$\mathcal{L} = \sum_{i \in \{1,T\}} \lambda_i L_i \tag{6.1}$$

In our case, we have two partial cost function:

- The cost function used for the first task (spike segmentation) is cross entropy, L_{T_1} (Equation 6.2), which calculates the negative log likelihood of the predicted class $predicted_j^1$:

$$L_{T_1} = -\sum_{j=1}^{2} GT_j^1 \log\left(predicted_j^1\right) \tag{6.2}$$

  where j represents the class value corresponding to spike versus background regions.

- The cost function used for SpikeCount optimisation, as discussed in Chapter 5, is pixel-wise $L2$ loss function to regress the per pixel density. Therefore, the cost function for the second task L_{T_2} is represented by Equation 6.3 where $D^p_{GT^2}$ is the density ground truth and $predicted^2$ is the predicted density for a certain pixel p in image X_i.

$$L_{T_2} = \sum_{p \in X_i} (D^p_{GT^2} - D^p_{predicted^2})^2 \tag{6.3}$$

Therefore, the global cost of training SpikeMulti is as following (Equation6.4):

$$\mathcal{L} = \lambda_1(-\sum_{j=1}^{2} GT_j^1 \log\left(predicted_j^1\right)) + \lambda_2(\sum_{p \in X_i} (D^p_{GT^2} - D^p_{predicted^2})^2) \tag{6.4}$$

6.1.2.2 Training hyperparameters

The hyperparameters need to be initialised before the training process starts. Then, the training algorithm learns new parameters as part of the learning process [118]. A summary of the SpikeMulti training hyperparameters values used in our study are as following:

- Weight θ (parameters) /Bias initialisation: It is good practice when training any deep learning model from scratch to initialise the weights with random values and the bias with 0. There are many initialisation techniques that can

be used but we found He et al.'s [51] technique achieved the optimal results when training from scratch. This technique generates a mean centred normal distribution with standard deviation σ equal to $\sqrt{2/n_l}$ where n_l is the number of inputs in a certain layer l.

- Dropout rate probability: This parameter serves as a regulariser to reduce the model overfitting [118]. It determines how many units can be deactivated randomly for every training iteration in a certain layer. In our model, two dropout layers are added, with a value of 0.5 for p, which represents the probability of keeping the unit activation or not. This is recommended when using layers with large numbers of units [118] such as FC6 and FC7. Therefore, dropout layers are added after every fully convolutional layer FC6 and FC7.

- Intermediate non-linearity unit: This is an essential component in any CNN that focuses on highlighting and emphasising the relevant features of the data and the task. As a default, we have selected Rectified Linear Unit (ReLu) for this parameter which is an element-wise thresholding operation that is applied on the output of the convolutional layer (resulting feature map) to suppress negative values: F(x)=max(0,x) where x is an element in the feature map.

- Epochs: This refers to the number of training iteration, and is different from one experimental setup to another.

- Optimisation algorithm: Since we have two different learning tasks to optimise simultaneously, we first utilised mini-batch stochastic gradient descent (SGD) optimisation method and then utilised mini-batch RMSprop optimising algorithm [52]. We have found that RMSprop optimising algorithm [52] has resulted in lower validation loss with a learning rate of 0.003 and mini-batch of 20 and the weight loss coefficients λ_1 and λ_1 of 1 because we aimed for each learning task loss to contribute equally in optimising SpikeMulti.

6.1.3 Training and validation of the architecture

Consistently with previous experiments, we have selected the CQ_2015 sequence for training SpikeCount and the CQ_2016 sequence as the validation set to observe if there is overfitting of the model. Also, as before, because of the high resolution of the images (2592 × 1944), we randomly sampled 450 sub-images with a size of 512 × 512 with their corresponding manual labels. This approach is to reduce computational complexity, as dealing with the Original image size would be unmanageable in the training process.

We have utilised an early stopping technique when training the model. Early stopping allows us to keep a record of the validation learning (e.g. cost and accuracy) for each learning epoch. It is a simple and inexpensive way to regularise the model and prevent overfitting as early as possible [63, 9]. We have selected the validation cost as the metric to observe for early stopping. The maximum epochs for observing the change in validation cost is 20 epochs. In other words, if the validation cost has not been decreased for 20 epochs, the model training will be stopped and the model weights resulting from the lowest validation cost are saved.

6.2 Multitask Infield Wheat Experimental Setup

Our experiments are as follows:

- **Experiment 1: Training SpikeMulti from Scratch** - with these set of experiments we try to understand how well the architecture does when trained from scratch in a conventional way. We initialise the model weights with He et al.'s [51] technique as discussed in the SpikeMulti Training subsection 6.1.2.2. These model training ran for 35 epochs.
- **Experiment 2: Training SpikeMulti by loading ImageNet learnt weights** - In addition to training SpikeMulti from scratch, we wanted to

investigate whether the Transfer Learning approach [103] can produce improvements of the validation accuracy. We loaded the pre-trained weights from the ImageNet challenge to the VGG16 and then trained the model with the same hyper-parameter settings described previously for 76 epochs.

6.3 Results

After training SpikeMulti on CQ_2015 and validating it on CQ_16 for both experimental setups discussed in last section, in this section we assess the performance of the model for both learning tasks: spike segmentation and spikelet counting. We have tested the model on the Original set containing the wheat crop images with background for both CQ_2016 and CQ_2017. We used a sliding-window approach which is discussed in Chapter 3.

First, we analyse the results of spike segmentation in general and in terms of growth stage breakdown for both experiments and compare it with the results of SpikeSEG performance when trained solely for spike segmentation, as presented in Chapter 4.

Then, we analyse the results of spikelet counting in general and in terms of growth stage breakdown for both experiments and compare it with the results of SpikeCount performance when trained solely for spikelets density estimation and tested on Original and Prediction sets, as presented in Chapter 5. We also report the breakdown of counting results for background and spike regions to investigate whether MTL have reduced the error metrics of background regions or not. We apply the multitask learning to the Original set, but also presented for comparison the results of the Prediction set since this uses a similar pipeline.

6.3.1 Investigating the importance of Multitask Learning on Spike Segmentation

Tables 6.1, 6.2, 6.3 and 6.4 summarise the performance of SpikeMulti results when tested for the first task, spike segmentation, on the Original sets CQ_2016 and CQ_2017 when trained from scratch and by loading ImageNet pre-trained weights. In addition, the tables compare the results of SpikeSEG performance when trained solely for spike segmentation in Chapter 4.

Tables 6.1, representing learning from scratch, and 6.2, representing Transfer Learning, show that training SpikeSEG solely has led to better segmentation across Mean Accuracy, Spike Accuracy and Spike IOU, but not Global Accuracy and Mean IoU. The improvement by using MTSL is of 2.39% (GA) and 0.07% (MIoU) when training from scratch and of 2.05% (GA) and 0.21% (MIoU) when loading ImageNet weights. Also, it is clear that training SpikeMulti by loading ImageNet weights have improved the results compared to training the model from scratch which is an established finding of Chapter 4.

Table 6.1: Quantitative results of segmentation for the CQ_2016 dataset comparing the performance of SpikeMulti model and SpikeSEG model when trained from scratch (by initialising both models the weights using He et al. [51] method) showing different evaluation metrics

Initialisation	GA	MA	Spike Accuracy	MIoU	Spike IoU
SpikeSEG-scratch	91.84%	**82.90**%	**70.22**%	73.43%	**55.98**%
SpikeMulti-scratch	**94.23**%	79.13%	59.78%	**73.50**%	53.18%
change as %	2.39%	-3.78%	-10.44%	0.07%	-2.80%

Table 6.2: Quantitative results of segmentation for the CQ_2016 dataset comparing the performance of SpikeMulti model and SpikeSEG model when loading pre-trained ImageNet parameters showing different evaluation metrics

Initialisation	GA	MA	Spike Accuracy	MIoU	Spike IoU
SpikeSEG-ImageNet	93.34%	**86.11%**	**75.85%**	77.63%	**62.75%**
SpikeMulti-ImageNet	**95.39**%	81.95%	64.73%	**77.84%**	60.64%
change as %	2.05%	-4.16%	-11.12%	0.21%	-2.11%

For CQ_2017, Tables 6.3 and 6.4 shows that using MTSL has significantly led to better segmentation across all metrics for both experimental setups. Particularly, Transfer Learning with MTSL led to substantial increases of 4.01% for Global

Accuracy, 8.06% for Mean Accuracy, 15.10% for spike accuracy, 10.47% for Mean IoU and 16.51% for Spike IoU.

Table 6.3: Quantitative results of segmentation for the CQ_2017 dataset comparing the performance of SpikeMulti model and SpikeSEG model when trained from scratch (by initialising both models the weights using He et al. [51] method) showing different evaluation metrics

Initialisation	GA	MA	Spike Accuracy	MIoU	Spike IoU
SpikeSEG-scratch	90.46%	73.15%	48.15%	66.69%	43.69%
SpikeMulti-scratch	**92.85%**	**73.62%**	**48.42%**	**68.47%**	**44.54%**
change as %	2.39%	0.47%	0.27%	1.78%	0.85%

Table 6.4: Quantitative results of segmentation for the CQ_2017 dataset comparing the performance of SpikeMulti model and SpikeSEG model when loading pre-trained ImageNet parameters showing different evaluation metrics

Initialisation	GA	MA	Spike Accuracy	MIoU	Spike IoU
SpikeSEG-ImageNet	92.47%	79.48%	60.72%	73.52%	55.35%
SpikeMulti-ImageNet	**96.48%**	**87.54%**	**75.82%**	**83.99%**	**71.85%**
change as %	4.01%	8.06%	15.10%	10.47%	16.51%

Table 6.5 illustrates the SpikeMulti spike segmentation performance breakdown for each growth stage when using Transfer Learning for CQ_2016 and provides comparisons with the results of SpikeSEG performance when trained solely for spike segmentation from Chapter 4. It is evident that MTSL has improved the performance for some metrics in early growth stages, in particular Booting and Heading. Notably, in the Heading growth stage, the Mean IoU and Spike IoU has increased significantly by 8.74% and 35.43% respectively. On the other hand, the performance has decreased for late growth stages, particularly, Flowering with a decrease of 17.17% for Spike accuracy and 10.27% for Spike IoU.

Table 6.5: Quantitative results comparing the segmentation performance of SpikeMulti-ImageNet and SpikeSEG-ImageNet for the CQ_2016 dataset reported for each growth stage

Growth Stage	Model	GA	MA	Spike Accuracy	MIoU	Spike IoU
Booting(GS45-47)	SpikeSEG	96.90%	**54.69%**	**11.88%**	49.75%	2.59%
	SpikeMulti	**99.38%**	54.32%	8.79%	**53.11%**	**6.84%**
change as %		2.48%	-0.37%	-3.09%	3.37%	4.25%
Heading(GS51-59)	SpikeSEG	92.94%	79.21%	**63.17%**	65.91%	39.22%
	SpikeMulti	**97.07%**	**79.64%**	60.12%	**74.65%**	**74.65%**
change as %		4.13%	0.43%	-3.05%	8.74%	35.43%
Flowering(GS61-69)	SpikeSEG	93.12%	**86.24%**	**75.68%**	**78.92%**	**65.77%**
	SpikeMulti	**93.92%**	78.85%	58.51%	74.46%	55.50%
change as %		0.80%	-7.39%	-17.17%	-4.46%	-10.27%
Grain filling (GS71-73	SpikeSEG	93.17%	**88.27%**	**80.05%**	81.09%	**70.33%**
	SpikeMulti	**95.16%**	86.28%	73.60%	**82.05%**	69.54%
change as %		1.99%	-1.99%	-6.45%	0.96%	-0.80%

Moving to CQ_2017, Table 6.6 presents a similar comparison. MTSL has this time improved the performance for some metrics in some growth stages. In particular, in the Booting growth stage where the MA and Spike accuracy has increased significantly by 16.98% and 33.38%. Also, the Mean IoU and Spike IoU has increased significantly by 18.48% and 36.11% respectively. This indicates that using both Transfer Learning and MTSL, which incorporates shared knowledge learnt from the second learning task, has a positive impact on improving the spike segmentation for a challenging sequence such as CQ_2017.

Table 6.6: Quantitative results comparing the segmentation performance of SpikeMulti and SpikeSEG for the CQ_2017 dataset reported for each growth stage

Growth Stage	Model	GA	MA	Spike Accuracy	MIoU	Spike IoU
Booting (GS45-47)	SpikeSEG	98.84%	67.14%	35.01%	57.78%	16.72%
	SpikeMulti	**99.69%**	**84.12%**	**68.39%**	**76.26%**	**52.83%**
change as %		0.85%	16.98%	33.38%	18.48%	36.11%
Heading (GS51-59)	SpikeSEG	94.02%	81.03%	64.61%	73.31%	53.03%
	SpikeMulti	**97.20%**	**87.42%**	**75.76%**	**82.81%**	**68.59%**
change as %		3.18%	6.39%	11.15%	9.50%	15.56%
Flowering (GS61-69)	SpikeSEG	91.09%	80.87%	63.65%	74.48%	59.20%
	SpikeMulti	**96.12%**	**89.60%**	**80.10%**	**86.00%**	**76.43%**
change as %		5.04%	8.74%	16.46%	11.52%	17.23%
Grain Filling (GS71-73)	SpikeSEG	87.54%	75.62%	52.65%	68.07%	50.42%
	SpikeMulti	**93.17%**	**82.94%**	**66.67%**	**78.32%**	**64.44%**
change as %		5.63%	7.32%	14.02%	10.25%	14.02%

From the above results, it is clear that employing ImageNet Transfer Learning has a positive effect on improving performance for both CQ_2016 and CQ_2017 datasets. As in Chapter 4 to further verify this finding, we present Precision-Recall curves in Figure 6.2 for each growth stage for the CQ_2016 and CQ_2017 datasets. The left most sub-figures show two graphs that represent the Precision-Recall curves of the models trained from scratch, whereas the right most graphs represent the curves after loading ImageNet parameters. The top two graphs refer to the 2016 validation dataset, whereas the bottom graphs present results for the 2017 dataset. Although there is relatively little improvement in Booting stage, it is noticeable that the SpikeMulti-Transfer Learning produces a "lift" effect on the Precision-Recall curve and significant lift in Heading, Flowering and Grain filling curves in CQ_2016. A significant lift on all curves for CQ_2017 (bottom right) including Booting growth stage.

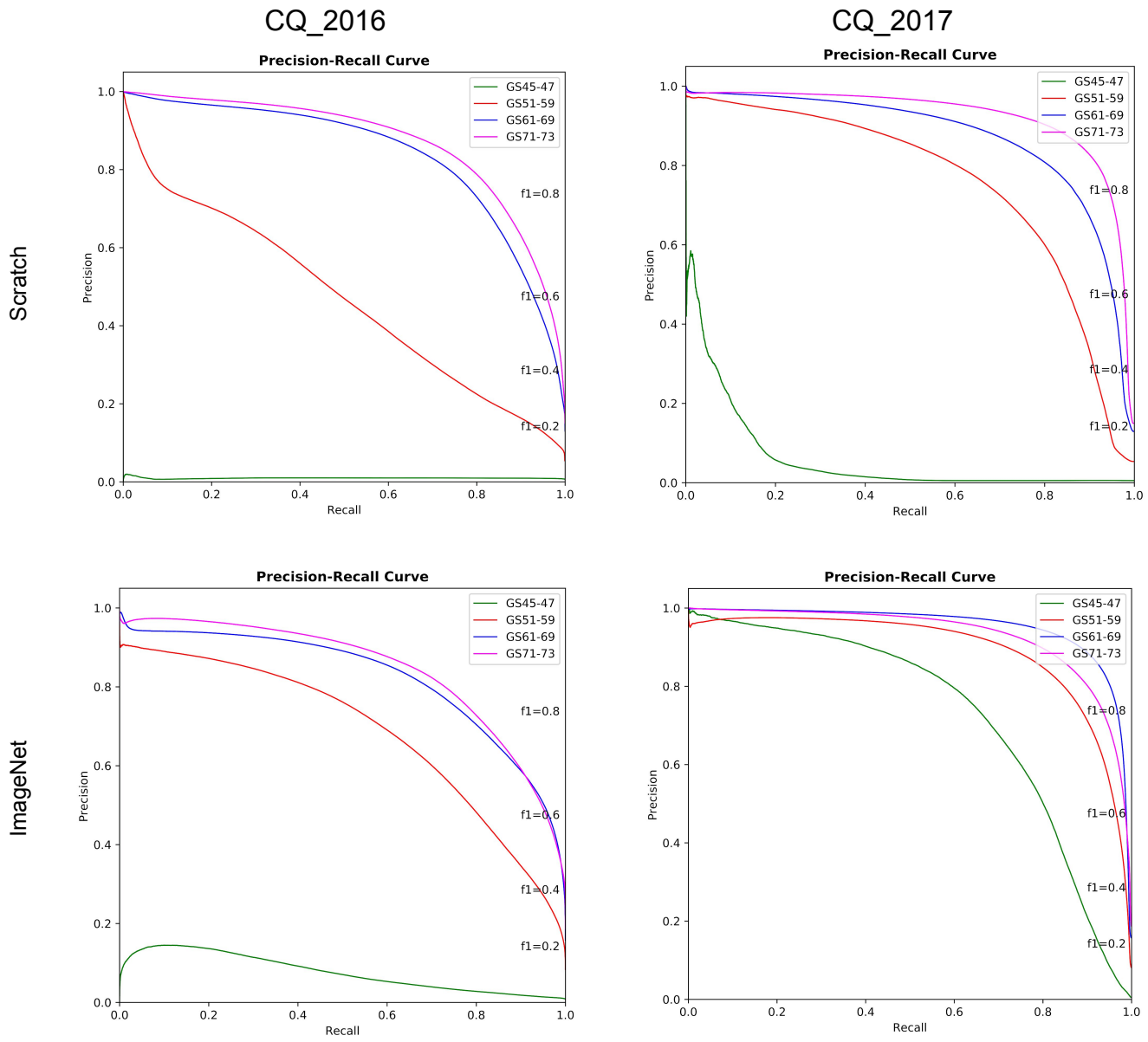

Figure 6.2: Precision-Recall curves showing the segmentation performance of SpikeMulti at different monitored growth stages for CQ_2016 (first column) and CQ_2017 (second column) when training from scratch (first row) and training transfer learning by loading pre-trained ImageNet parameters (second row).

Compared to Figure 4.2 when using Transfer Learning with SpikeSEG in Chapter 4, SpikeMulti and Transfer Learning has lifted Booting, Heading and Grain filling curves in CQ2016, but in particularly Heading and Grain filling curves. A significant lift to all growth stage curves is noted particularly in Booting in CQ_2017.

6.3.2 Investigating the importance of Multitask Learning on Spikelet Counting

Now we turn our attention to Tables 6.7 and 6.8 summarising the performance of SpikeMulti which was trained on two tasks simultaneously: spike segmentation and spikelets numbers estimation. We only test the MTL model on Original set for obvious reason; testing on the Prediction set is not required because it is segmented using SpikeSEG presented in Chapter 4. The tables show, in the final two rows, the results of testing SpikeMulti on the Original sets CQ_2016 and CQ_2017 in relation to the second task, i.e. spikelet count estimation. The first of those two rows relates to the results of training the SpikeMulti from scratch; the second relates to loading ImageNet pre-trained weights (Transfer Learning). In addition, we reproduce the comparable results from Chapter 5 for the Original and Prediction sets. We compare the performance of SpikeMulti against the Original set to understand how MTL performs for a hard problem. We also compare to the Prediction set because in this set the images were generated by applying SpikeSEG and then tested via SpikeCount; therefore it follows the same pipe-line as the SpikeMulti and it will show if there is any advantage in learning both task simultaneously.

For CQ_2016 dataset the results are shown in Table 6.7, and we can see that loading pretrained ImageNet weights when looking at MTL has a positive impact when comparing against training the SpikeMulti from scratch. The MSE, MAE and SMAPE have improved by 52.97%, 56.73% and 57.59% respectively.

For the Original set, from Table 6.7, it is clear that MTL has improved significantly the spikelet number estimation. For example, in terms of comparing

Table 6.7: The Mean Square Error, Mean Absolute Error, and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets highlighting the effect of multitask learning and comparing to previous experiments (displayed as rows) for Original and pre-segmenting (predicted) images (displayed in columns) on CQ_2016

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	2709.92	2566.43	89.43	1904.82	1789.72	64.46
Loading ACID Weights	3864.54	3488.43	**55.43**	1718.22	1590.85	47.99
Freezing Conv1 Block	**2557.82**	**2344.70**	86.10	1647.86	1531.27	44.65
Freezing Conv1-2 Blocks	3079.62	2916.16	89.16	2184.802	2063.195	69.733
Freezing Conv1-3 Blocks	2757.982	2607.525	92.835	2506.520	2375.151	86.856
Freezing Conv1-4 Blocks	2635.776	2504.633	93.198	2413.221	2285.898	95.545
Freezing Conv1-5 Blocks	2384.91	2257.61	96.895	2395.447	2269.008	92.399
Augmentation-Scratch	-	-	-	2118.51	1877.35	38.08
Augmentation-ACID	-	-	-	**1364.22**	**1209.75**	**34.24**
			SpikeMulti			
Training From scratch	1822.67	1729.59	67.37			
Loading ImageNet Weights	**857.20**	**748.34**	**28.57**			

SpikeMulti-scratch with the best approach of SpikeCount-Original, the MSE, MAE and SMAPE have improved by 52.97%, 56.73% and11.94% respectively. In terms of comparing SpikeMulti-ImageNet to the best approach of SpikeCount-Original, the MSE, MAE and SMAPE have improved substantially by 66.49%, 68.08% and 26.86% respectively. As for comparing SpikeMulti-scratch to the Prediction set results for CQ_2016, there is no improvement when training SpikeMulti from scratch. However, a significant improvement is clear when comparing to employing Transfer Learning in the context of MTL with a decrease of 37.17%, 38.14%, and 21.23% for MSE, MAE and SMAPE respectively.

Figures 6.3 and 6.4 present the breakdown of errors (in background and spike regions) from employing MTL and other experimental setups discussed in Chapter 5 for Original and Prediction sets of CQ_16 and CQ_17. The new MTL experiments as shown as columns E5.1 and E5.2 in the graphs. Looking at Figure 6.3, Original set, we can see the impact of simultaneously training the model on both tasks on reducing the background errors based both on training from scratch (E5.1) and using Transfer Learning (E5.2). It is also worth noting that the approach is able to maintaining low spike region errors. For example, employing MTL with Transfer Learning (E5.2) has led to significant reduction of background based errors with an MSE of 123.32 and MAE of 115.07. In terms of spike region errors, the

same experiments (E5.2) led to good results with a MSE of 733.88 and small increase in MAE from E2, with a value of 633.28. Looking at the Prediction set, background errors have been minimised when utilising MTL, but for spike region errors a combination of data augmentation and Transfer Learning does slightly better.

For CQ_2017 dataset reported in Table 6.8, we can see that loading pretrained ImageNet weights (i.e. Transfer Learning) has a positive impact on results for MTL. The MSE, MAE and SMAPE have improved by 56.91%, 61.54% and 62.19% respectively with respect to training MTL from scratch. Then, we compare the best performance metrics of SpikeMulti with all model variations for Original and Prediction sets. For the Original set, it is clear that MTL has improved significantly the spikelet estimation. For example, in terms of comparing SpikeMulti-scratch with the best approach of SpikeCount-Original, the MSE, MAE and SMAPE have improved by 26.05%, 25.60% and 19.92% respectively. Moreover, in terms of comparing SpikeMulti-ImageNet to the best approach of SpikeCount-Original, the MSE, MAE and SMAPE have significantly improved by 65.82%, 69.38% and 30.07% respectively. As for comparing SpikeMulti-scratch to the Prediction set results of CQ_2017, there is no improvement when training from scratch. However, a significant improvement is clear when comparing SpikeMulti-ImageNet to the best approach for SpikeCount-Prediction with a decrease of 28.69%, 34.32% and 1.51 for MSE, MAE and SMAPE respectively.

As we saw for CQ_2016, looking at Figure 6.4 which represents breakdown of errors for CQ_2017, for the Original set we can see the impact of MLT on reducing the background errors both when training from scratch and using Transfer Learning. The approach also maintains low spike region based errors. For example, multitask learning with Transfer Learning (E5.2) has led to significant reduction in background errors with an MSE of 116.96 and MAE of 109.26. For spike regions, the same experiments (E5.2) led to a very slight increase in errors with MSE of 1108.77 and MAE of 874.57. Comparing to the Prediction set, it is evident that

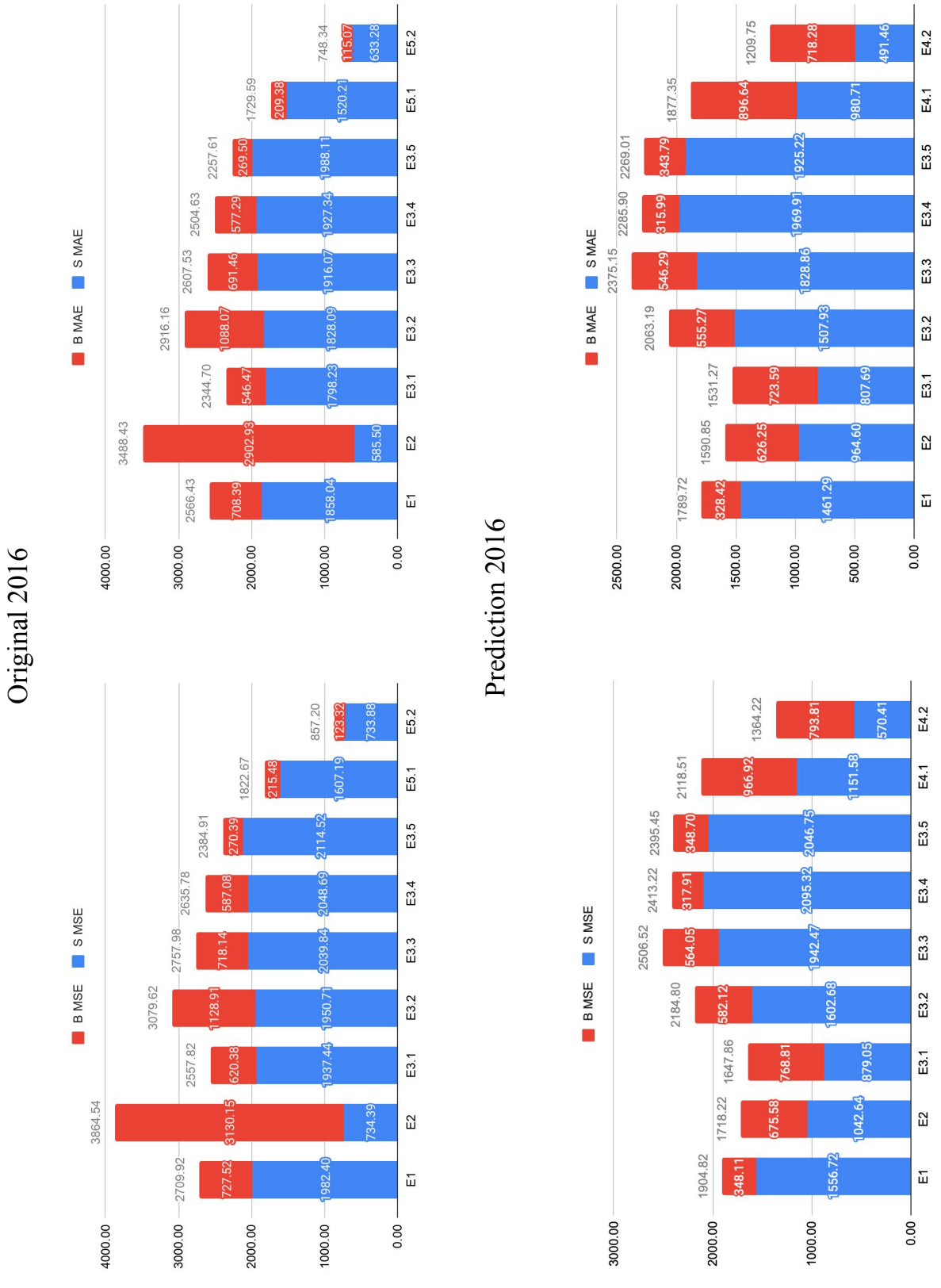

Figure 6.3: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of CQ_2016 sequence. E5.1/E5.2 report on the MTL.

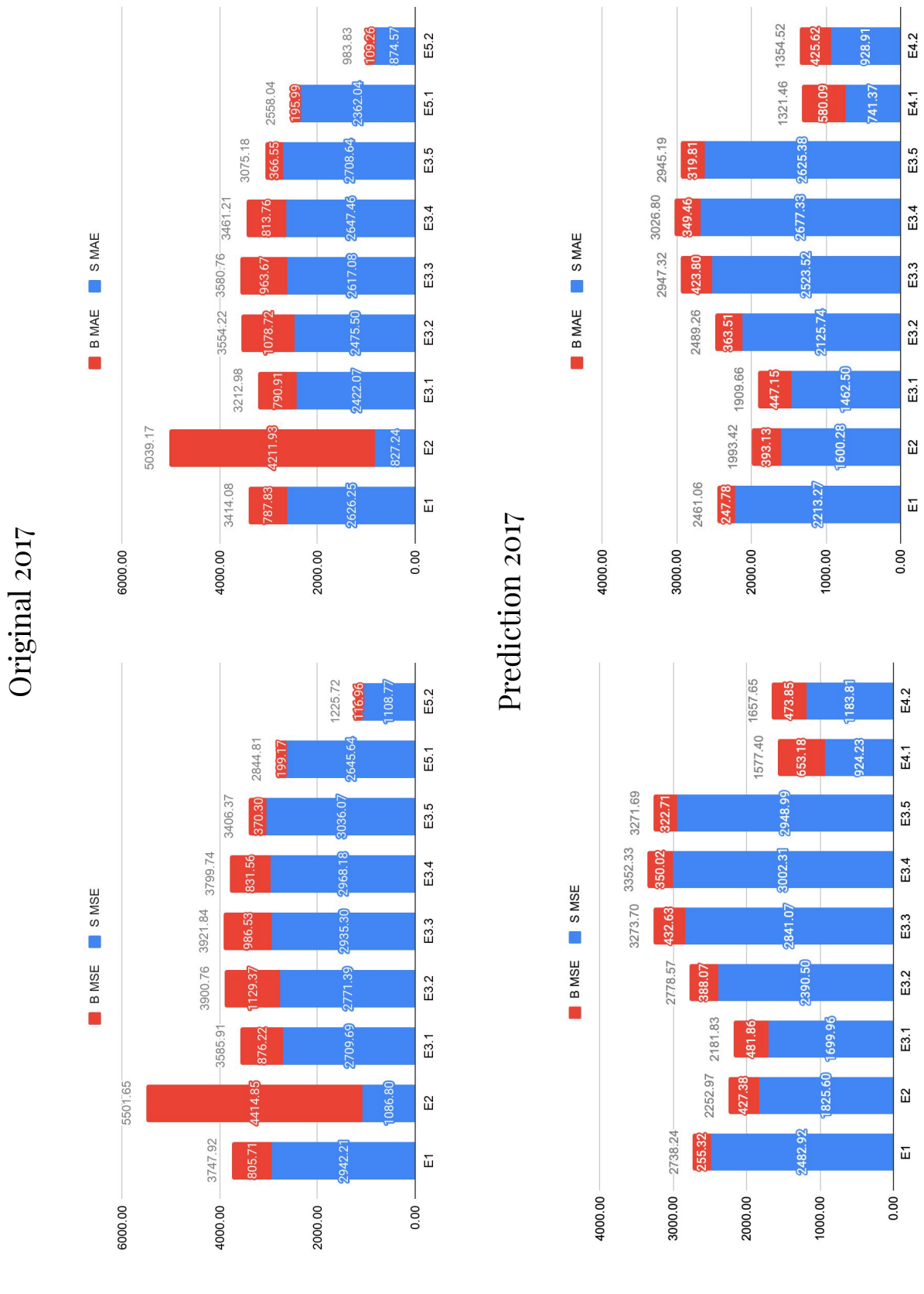

Figure 6.4: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of CQ_2017 sequence. E5.1/E5.2 report on the MTL.

both background and spike region errors have been minimised when utilising MTL.

Table 6.8: The Mean Absolute Error, Mean Square Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets highlighting the effect of utilising multitask learning and compared to previous experiments (displayed as rows) for Original and pre-segmenting (predicted) images (displayed in columns) on CQ_2017

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	3747.92	3414.08	93.90	2738.24	2461.06	73.67
Loading ACID Weights	5501.65	5039.17	**60.46**	2252.97	1993.42	54.49
Freezing Conv1 Block	**3585.91**	**3212.98**	86.44	2181.83	1909.66	51.35
Freezing Conv1-2 Blocks	3900.76	3554.22	88.48	2778.57	2489.26	71.62
Freezing Conv1-3 Blocks	3921.84	3580.76	93.60	3273.70	2947.32	88.03
Freezing Conv1-4 Blocks	3799.74	3461.21	94.72	3352.33	3026.80	95.54
Freezing Conv1-5 Blocks	3406.37	3075.18	97.26	3271.69	2945.19	92.72
Augmentation-Scratch	-	-	-	**1577.40**	**1321.46**	**31.90**
Augmentation-ACID	-	-	-	1657.65	1354.52	37.74
			SpikeMulti			
Training From scratch	2844.81	2558.04	80.38	-	-	-
Loading ImageNet Weights	**1225.72**	**983.83**	**30.39**	-	-	-

After comparing the results of SpikeMulti on the Original set to the prediction of SpikeCount on Original and Prediction sets, it is clear that SpikeMulti has outperformed SpikeCount when tested on Original set and Prediction set for both growing seasons in terms of combined errors when merging the background error with spike region error. It has also minimised the background errors notably and led to competitive spike regions errors.

Now, we want to briefly compare the SpikeMulti counting results to the prediction of SpikeCount on the Optimal set as that represents the upper bound of efficacy in predicting spikelet numbers. According to Chapter 5, it represents the best results obtained, however, it relies on a perfect segmentation mask. On the other hand, SpikeMulti can be realistically deployed in real time without requiring a segmentation mask and can produce reasonable outcomes. For fairness, we will only compare the results of spike region errors. For CQ_2016, the best results for SpikeCount were an MSE, MAE and SMAPE of 315.961, 270.288 and 8.87%, and those were obtained when freezing the first encoder, conv1, after loading the ACID weights. The difference in errors is 417.92, 362.99 and 19.70 % with respect to SpikeMulti measures. For CQ_2017, the best results for SpikeCount were an MSE, MAE and SMAPE of 667.34, 544.32, 16.32% giving a difference of 441.43,

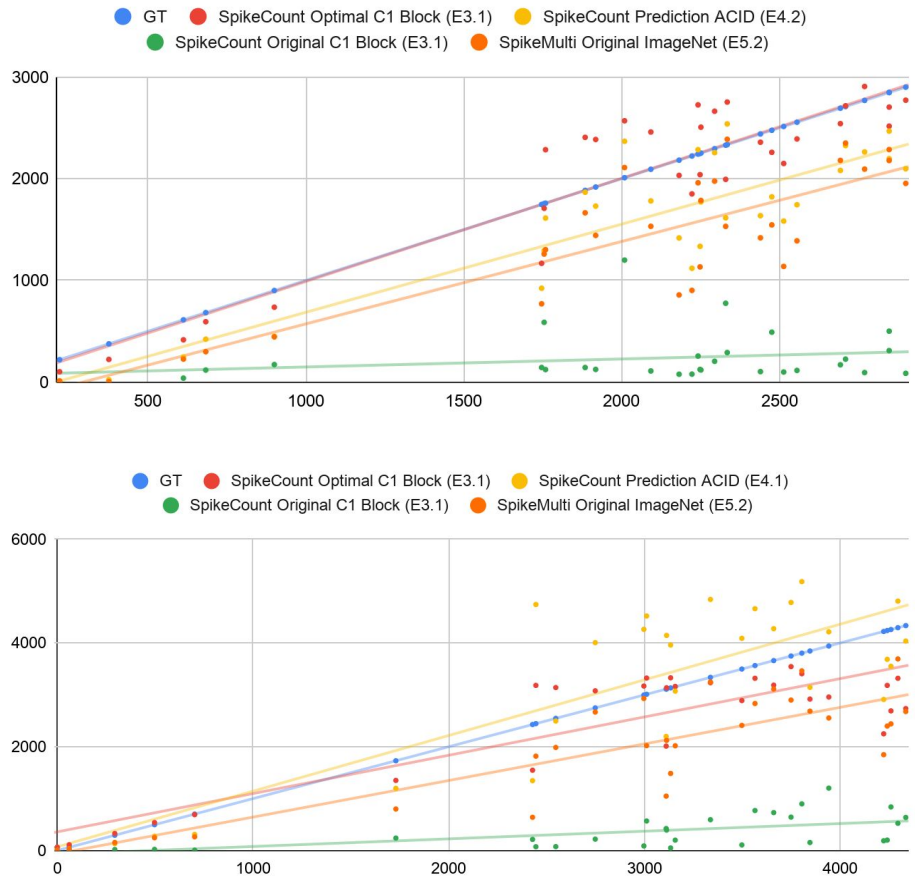

Figure 6.5: The correlation graphs showing predicted counts for SpikeMulti (orange) and the ground truth (blue). For SpikeCount, we present the best experimental setup for Original (green), Optimal (red) and Prediction (yellow) sets; the first graph represents 2016 growing season and second graph represents 2017 growing season

330.25, 14.07 from SpikeMulti measures. This shows that SpikeMulti has provided comparable results to the best SpikeCount for the Optimal set, which indicates a very good performance. Results are particularly competitive for CQ_2017 and that may be due to the improvements on segmentation performance.

Table 6.9: The Mean Squared Error, Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing the best performance of testing SpikeCount on Optimal set and the Spike region errors of testing SpikeMulti on Original set (displayed as rows) on CQ_2016 and CQ_2017 images (displayed in columns)

	CQ_2016			CQ_2017		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
SpikCount-Optimal	**315.96**	**270.29**	**8.87**	**667.34**	**544.32**	**16.32**
SpikeMulti	733.88	633.28	28.57	1657.65	874.57	30.39
Absolute Diff.	417.92	362.99	19.70	441.43	330.25	14

Figure 6.5 illustrates the correlation graphs for the various models, SpikeCount and SpikeMulti, and the ground truth counts. It reports the best experimental setup for Original, Optimal and Prediction sets. Each time counts refer only to the spike region area. The first graph represents 2016 growing season and second graph represents 2017 growing season. For CQ_2016, looking at the regression lines, SpikeCount-Optimal counts (red) are very well correlated with the ground truth (blue); the second best correlation comes from SpikeCount-Prediction (yellow) and then from SpikeMulti (orange) counts. For CQ_2017, both SpikeMulti counts (orange) and SpikeCount-Optimal (red) are well correlated. The SpikeCount-Prediction (yellow) is showing slight over-counting but can be considered the best correlation to the ground truth. Lastly, SpikeCount-Original (green) for both seasons has the lowest correlation with the ground truth.

6.3.3 Phenotypic analysis Spikelets Estimation for Growth Stages

In this section, we analyse spikelet estimation results from the perspective of wheat growth stages after testing SpikeMulti and compare it to previous results produced by experimental setups discussed in Chapter 5.

6.3.3.1 G45-47: Booting

First, we start the analysis by looking at the Booting growth stage and the results of SpikeMulti spikelet for CQ_2016 (Table 6.10) and CQ_2017 (Table 6.11). Those are reported for two sets: Original and Prediction. Table 6.10 shows SpikeMulti performance evaluation when tested on Booting growth stage images in CQ_2016. In addition, Figure 6.6 provides the quantitative errors from the spike and background regions separately.

Table 6.10 shows that MTL has resulted in the lowest errors across all experimental setups for Original and Prediction sets. Particularly, using both MTL and Transfer

Table 6.10: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Booting** growth stage of CQ_2016 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	673.17	662.24	99.80	**555.07**	**543.19**	98.55
Loading ACID Weights	1942.26	1925.73	**99.67**	741.20	719.47	97.13
Freezing Conv1 Block	**588.37**	**576.78**	99.92	817.59	794.43	97.09
Augmentation-Scratch	-	-	-	964.15	936.96	**95.10**
Augmentation-ACID	-	-	-	820.44	797.38	95.64
			SpikeMulti			
Training From scratch	530.29	519.02	99.49	-	-	-
Loading ImageNet Weights	**325.29**	**313.73**	**95.33**	-	-	-

Learning has the lowest MSE of 325.29 and MAE 313.37. Also, it produces the lowest relative errors percentage (SMAPE) of 95.33% compared to the results of Original set and a slight increase of 0.23% compared to using augmentation in the Prediction set.

Figure 6.6 illustrates that MTL has resulted in the lowest background errors across all experimental setups for Original and Prediction sets of the Booting growth stage. Particularly, utilising both MTL and Transfer Learning (E5.2) has minimised the background based errors significantly with an MSE of 23.12 and MAE of 22.49. The experimental setup for MTL and Transfer Learning (E5.2) has also led to the lowest spike region errors for Original set with MSE of 302.16 and MAE of 291.24 and slight increase of 4.2% for MSE and 3.83% for MAE when compared to spike region errors resulting from only augmentation (E4.1) in the Prediction set.

For illustration purposes, Figure 6.7 shows comparisons of visualisation of the spikelets density and segmentation maps predicted using SpikeMulti and density map predicted using SpikeCount and predicted segmentation map using SpikeSEG for one example in Original set of the Booting stage of 2016 image series.

Table 6.11 shows SpikeMulti performance evaluation when tested on images in Booting growth stage in the CQ_2017 sequence. Also, Figure 6.9 provides the quantitative errors from the spike and background regions separately.

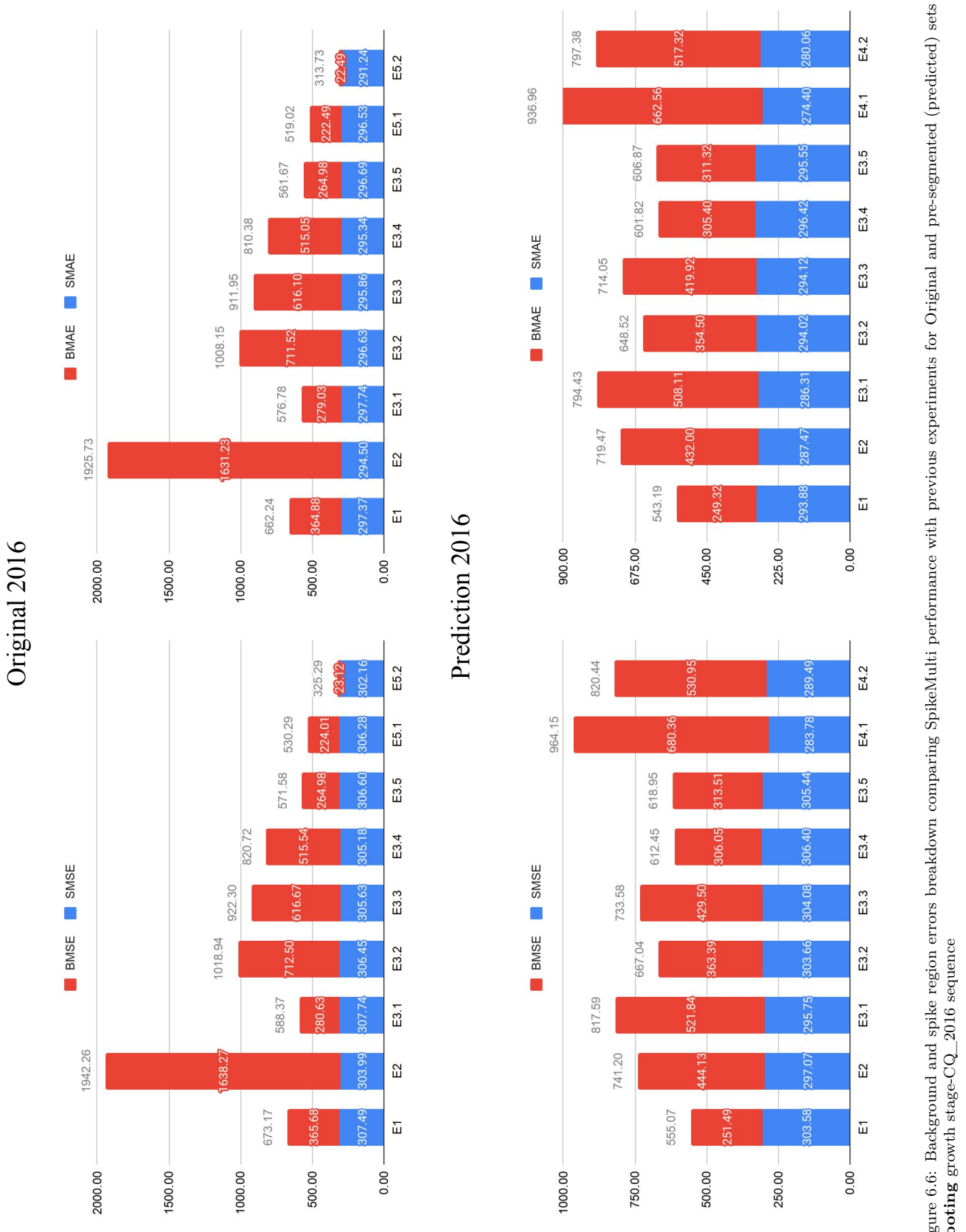

Figure 6.6: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Booting** growth stage-CQ_2016 sequence

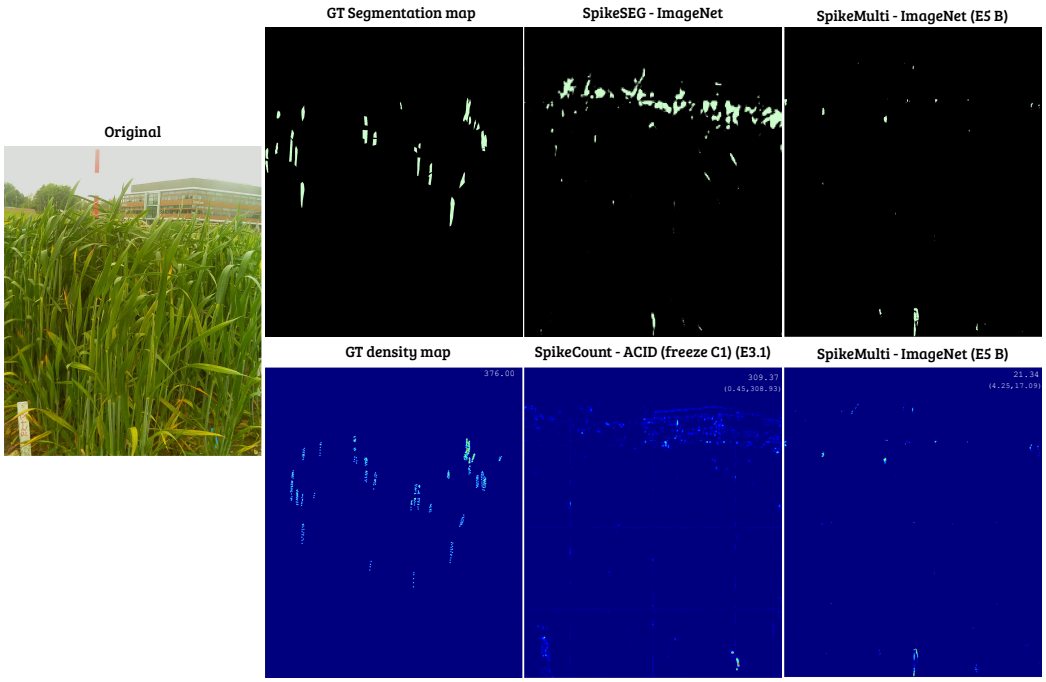

Figure 6.7: Visualisation of the spikelet segmentation and density maps for the **Booting** stage of 2016 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

Table 6.11: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Booting** growth stage of CQ_2017 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	**1165.53**	**1072.40**	99.14	407.63	329.50	92.39
Loading ACID Weights	5944.61	5832.97	**97.56**	405.26	330.83	89.02
Freezing Conv1 Blocks	1459.11	1317.07	98.97	458.86	385.48	89.54
Augmentation-Scratch	-	-	-	431.42	382.61	**80.10**
Augmentation-ACID	-	-	-	**386.92**	**327.46**	81.26
			SpikeMulti			
Training From scratch	463.06	376.51	97.56	-	-	-
Loading ImageNet Weights	**165.86**	**117.09**	**65.23**	-	-	-

Table 6.11 shows that MTL has resulted in a significant decrease in errors across all experimental setups for Original set of the Booting growth stage. Particularly, utilising both MTL and Transfer Learning has the lowest MSE of 165.86 and MAE 117.09. Also, it produced the lowest relative error percentage (SMAPE) of 65.23% when comparing to the results of Original and Prediction sets.

Figure 6.9 illustrates that for CQ_2017, MTL has resulted in the lowest background errors across all experimental setups for the Original set at the Booting growth stage. As before, using both MTL and Transfer Learning (E5.2) has minimised the background based errors significantly with an MSE of 36.45 and MAE of 32.35. It has also led to the lowest spike region errors for the Original set with MSE of 129.41 and MAE of 84.75 and a slight increase of 4.45% for MSE and 1.9% for MAE when compared to spike region errors obtained with augmentation in the Prediction set.

Figure 6.8 illustrates comparisons of visualisation of the spikelets density and segmentation maps predicted using SpikeMulti and density map predicted using SpikeCount and predicted segmentation map using SpikeSEG for one example in Original set of the Booting stage of 2017 image series.

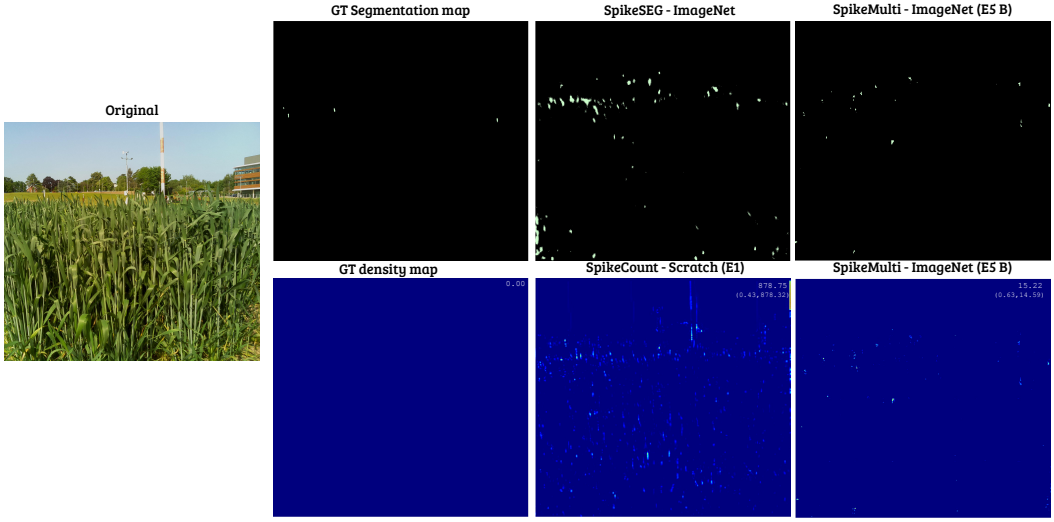

Figure 6.8: Visualisation of the spikelet segmentation and density maps for the **Booting** stage of 2017 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

6.3.3.2 G51-59:: Heading

The analysis of Heading growth stage images with SpikeMulti for CQ_2016 and CQ_2017 are reported in Table 6.12 and Table 6.13 respectively for two sets:

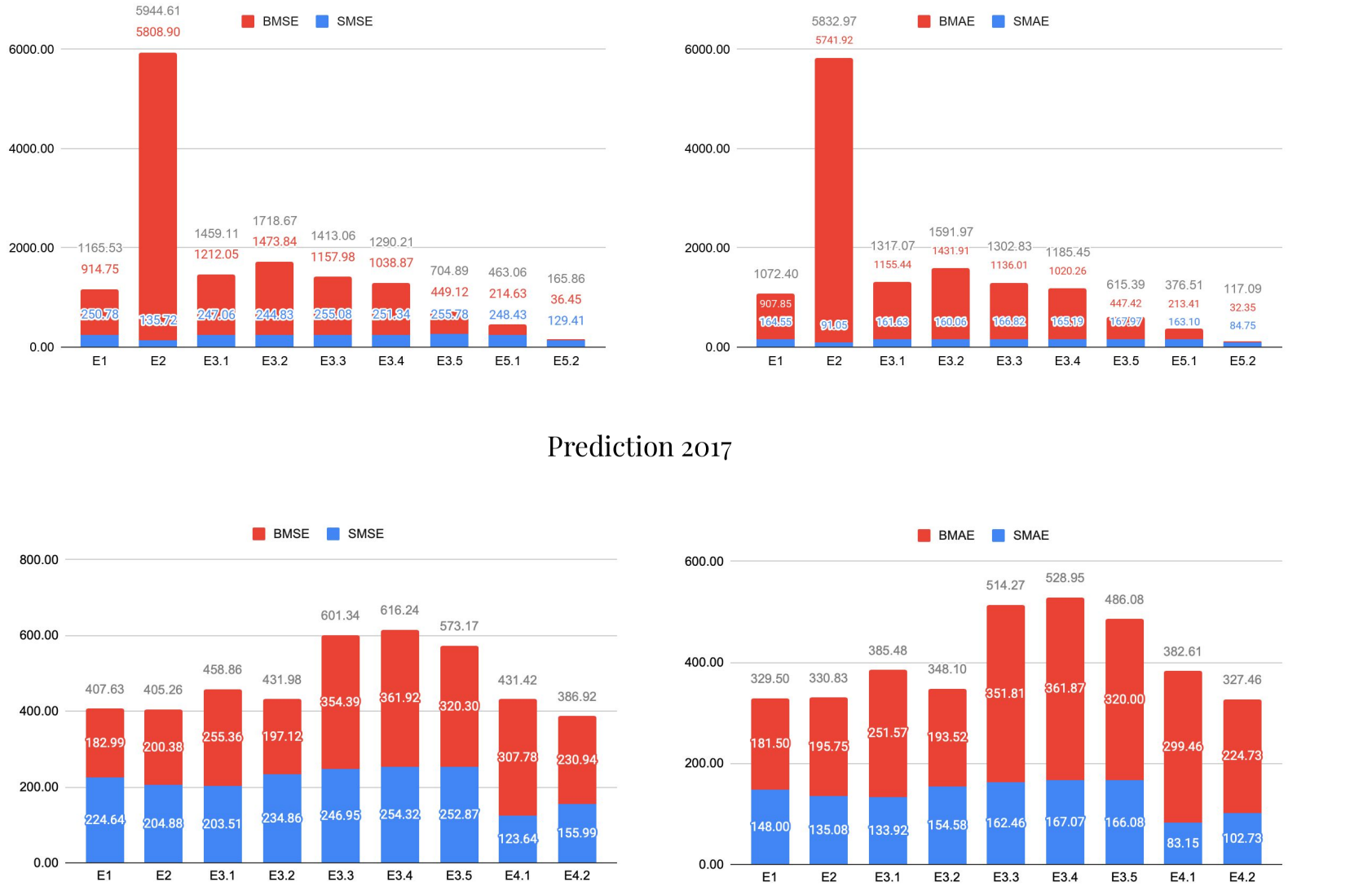

Figure 6.9: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Booting** growth stage-CQ_2017 sequence

Original and Prediction.

Table 6.12: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Heading** growth stage of CQ_2016 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	**2212.95**	**2070.73**	88.35	1718.51	1601.91	72.91
Loading ACID Weights	5135.41	4839.04	**70.46**	1912.97	1806.62	64.38
Freezing Conv1 Blocks	2270.61	2098.48	82.67	2040.50	1940.01	67.59
Augmentation-Scratch	-	-	-	**1670.28**	**1591.04**	**46.09**
Augmentation-ACID	-	-	-	1844.85	1736.82	53.98
			SpikeMulti			
Training From scratch	1672.56	1559.61	83.79	-	-	-
Loading ImageNet Weights	**920.16**	**834.49**	**38.58**	-	-	-

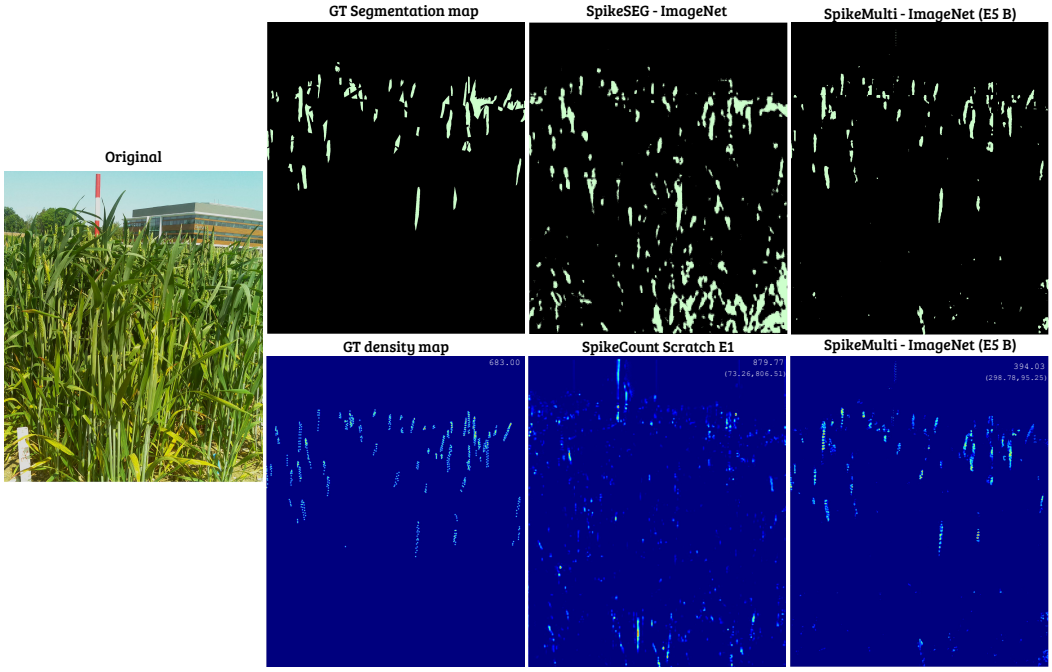

Figure 6.10: Visualisation of the spikelet segmentation and density maps for the **Heading** stage of 2016 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

Table 6.12 shows SpikeMulti performance evaluation when tested on images from the Heading growth stage in CQ_2016 sequence. In addition, Figure 6.11 provides the quantitative errors for the spike and background regions separately.

Table 6.12 shows that MTL has resulted in the lowest errors across all experimental setups for Original set of the Heading growth stage. Again, MTL and Transfer Learning has the lowest MSE of 920.16 and MAE 834.49. Also, it produces the lowest relative error percentage (SMAPE) of 38.58% compared to the results of Original and Prediction sets.

Figure 6.10 illustrates visualisation of the spikelet density and segmentation maps predicted using SpikeMulti and SpikeCount and also the predicted segmentation map using SpikeSEG. These are given for one example Heading stage image in the Original set for the 2016 image series.

Consistently with other experiments, Figure 6.11 illustrated that MTL has resulted in the lowest background errors across all experimental setups for Original set of the Heading growth stage. MTL and Transfer Learning has minimised the background errors significantly with an MSE of 155.76 and MAE of 145.94. However, in terms of spike region errors, this experimental set up (E5.2) led to an increase of 35.0% for MSE and 21.7% for MAE when compared to utilising Transfer Learning from ACID (E2) for Original set. Also, it led to an increase of 58.0% for MSE and 57.7% for MAE when compared to spike region errors when using augmentation (E4) in the Prediction set.

Table 6.13 shows SpikeCount performance evaluation in CQ_2017 sequence. Also, Figure 6.13 provides the quantitative errors from the spike and background regions separately. In addition, Figure 6.12 illustrates comparisons of visualisation of the spikelets density and segmentation maps predicted using SpikeMulti and density map predicted using SpikeCount and predicted segmentation map using SpikeSEG for one example in Original set of the Heading stage of 2017 image series.

Again, Table 6.13 shows that for the Original set MTL has resulted in the lowest errors across all experimental setups for the Heading growth stage. MLT and Transfer Learning has the lowest MSE of 1849.83 and MAE of 1730.90 for the Original set. Also, it produced the lowest relative error percentage (SMAPE)

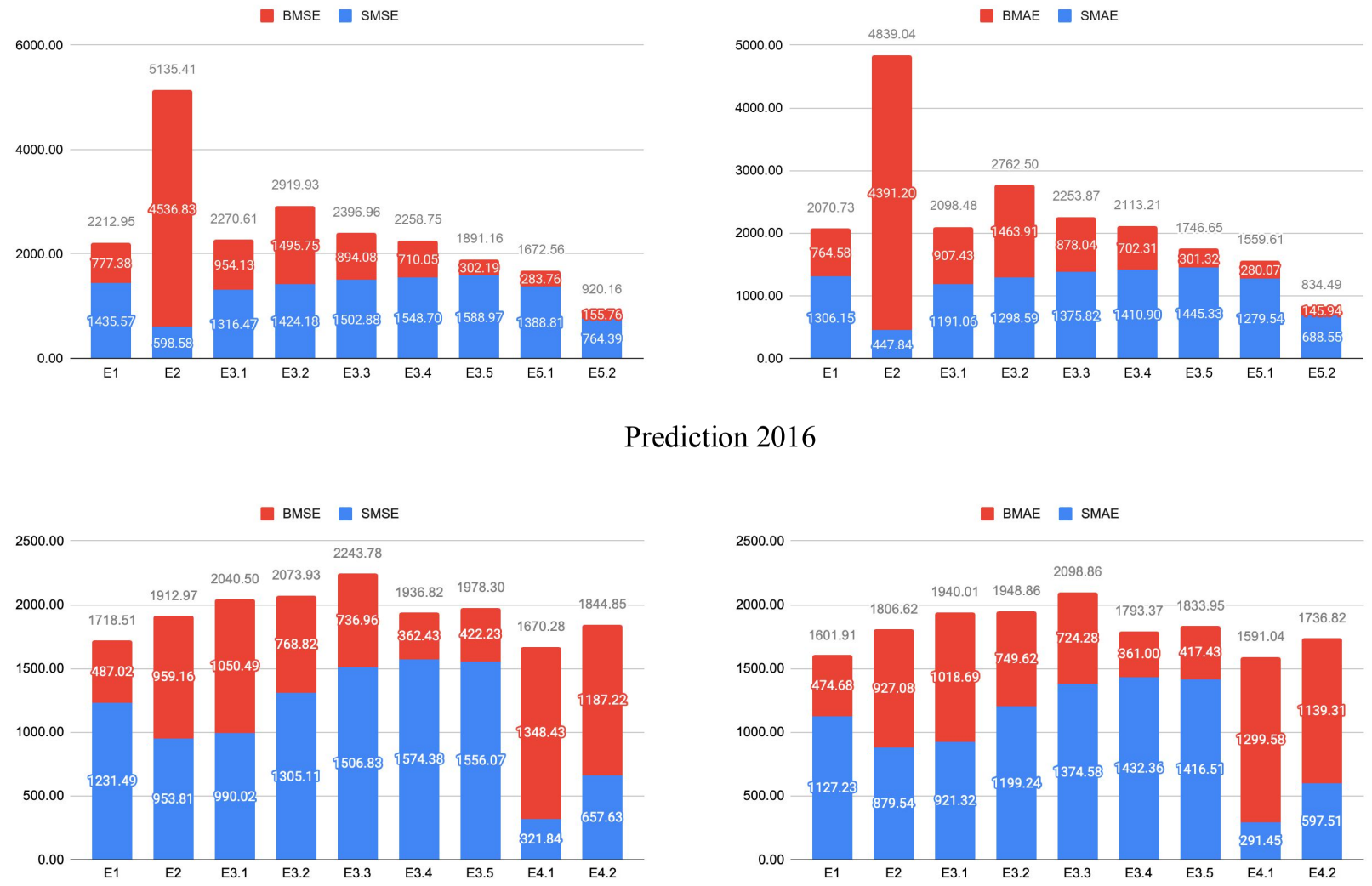

Figure 6.11: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Heading** growth stage-CQ_2016 sequence

Table 6.13: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Heading** growth stage of CQ_2017 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	**3788.10**	**3515.00**	93.83	3024.86	2795.54	77.00
Loading ACID Weights	5757.32	5441.61	**64.95**	2785.21	2587.53	62.90
Freezing Conv1 Blocks	3791.42	3496.96	85.38	2850.84	2650.28	63.25
Augmentation-Scratch	-	-	-	**1718.77**	**1575.63**	**33.39**
Augmentation-ACID	-	-	-	2380.83	2208.17	49.61
			SpikeMulti			
Training From scratch	3121.32	2887.84	84.93	-	-	-
Loading ImageNet Weights	**1849.83**	**1730.90**	**42.72**	-	-	-

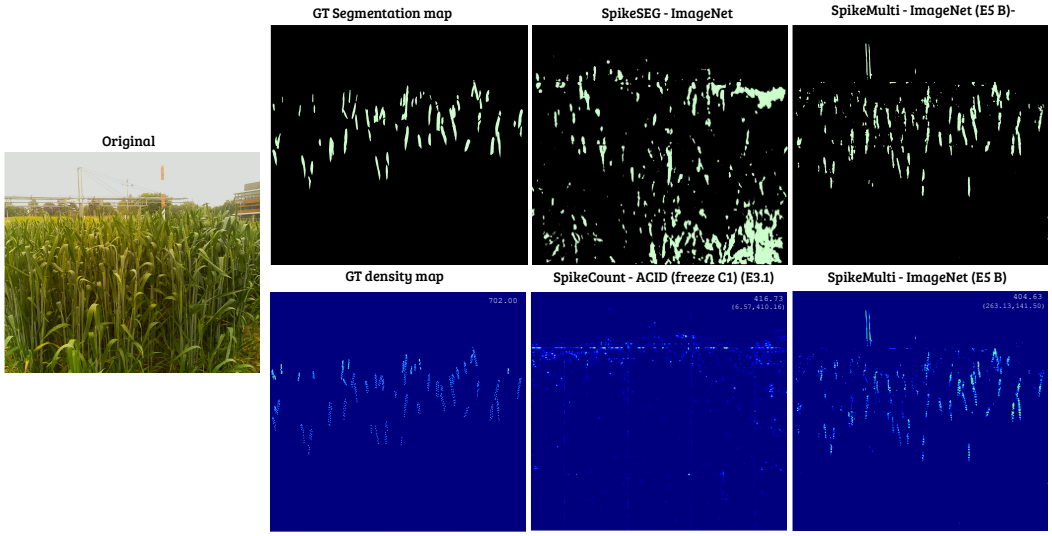

Figure 6.12: Visualisation of the spikelet segmentation and density maps for the **Heading** stage of 2017 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

of 42.72%. However, it has increased the errors by 7.08% for MSE, 8.97% for MAE and 9.33% for relative error percentage (SMAPE) when compared to using augmentation (E4.1) for the Prediction set.

Figure 6.13 illustrates the same as before, MTL results in the lowest background errors across all experimental setups when applied to the Original set and compared to the Prediction sets in the Heading growth stage. MLT and Transfer Learning (E5.2) has minimised the background based errors significantly with an MSE of

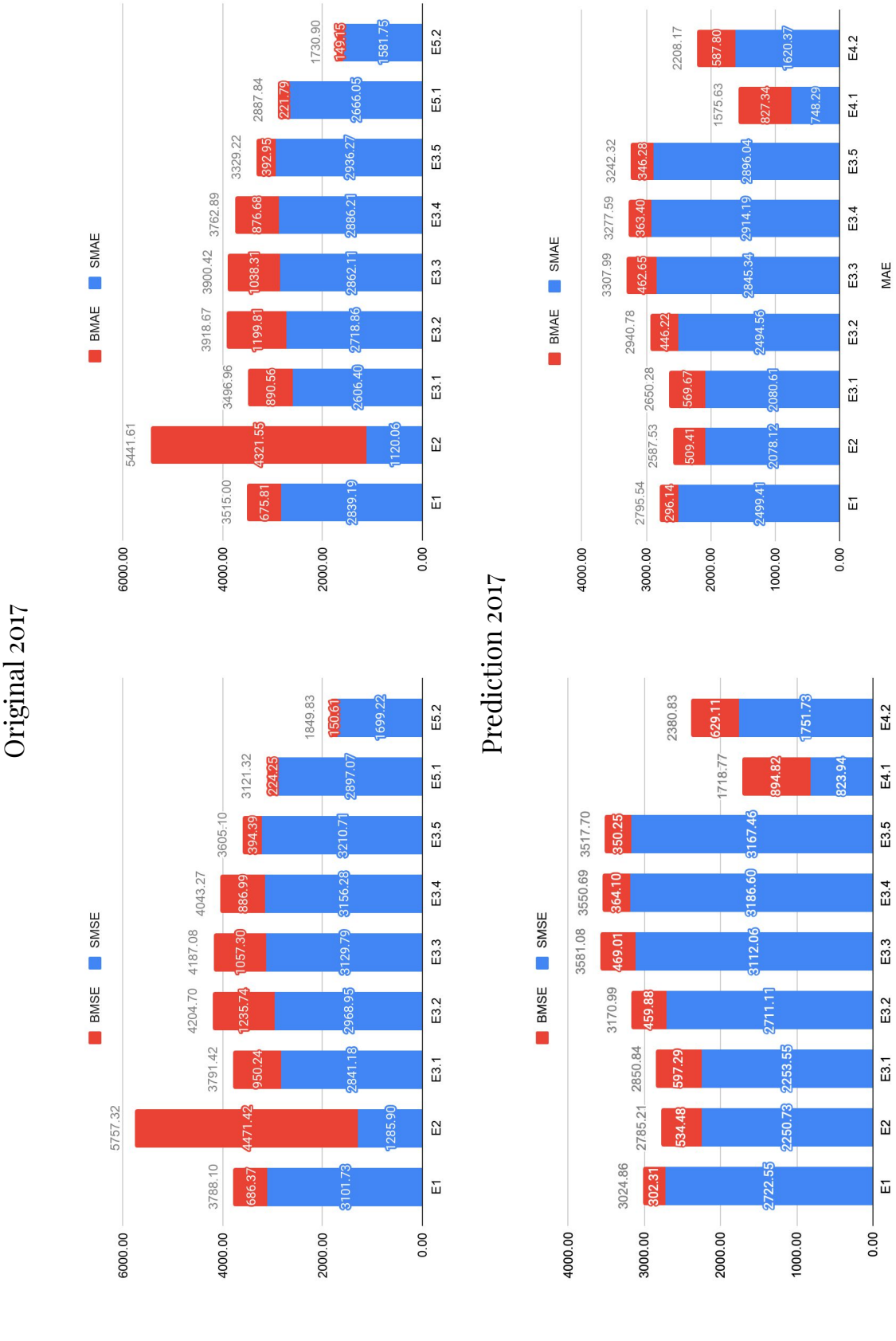

Figure 6.13: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Heading** growth stage-CQ_2017 sequence

150.61 and MAE of 149.15. However, for the spike region errors, it has led to an increase of 24.32% for MSE and 29.19% for MAE when compared to utilising Transfer Learning from ACID (E2) for the Original set. Also, it has led to an increase of 51.51% for MSE and 52.69% for MAE when comparing to spike region errors from augmentation in the Prediction set.

6.3.3.3 G61-69: Flowering

For the Flowering growth stage the results of SpikeMulti for spikelet estimation for CQ_2016 (Table 6.14) and CQ_2017 (Table 6.15) are reported for two sets: Original, and Prediction.

Table 6.14: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Flowering** growth stage of CQ_2016 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	2753.53	2709.59	88.20	1893.45	1846.88	57.43
Loading ACID Weights	3350.92	3066.16	**47.06**	1656.17	1546.81	38.85
Freezing Conv1 Blocks	**2523.45**	**2416.61**	85.05	1571.04	1466.18	34.20
Augmentation-Scratch	-	-	-	1972.68	1826.60	28.79
Augmentation-ACID	-	-	-	**1266.11**	**1133.29**	**24.44**
			SpikeMulti			
Training From scratch	1861.11	1828.79	60.03	-	-	-
Loading ImageNet Weights	**993.68**	**890.16**	**23.87**	-	-	-

Table 6.14 shows SpikeMulti performance evaluation when tested on CQ_2016 sequence. In addition, Figure 6.15 provides the quantitative errors from the spike and background regions.

Again consistently, Table 6.14 shows that MTL results in the lowest errors across all experimental setups. MTL and Transfer Learning has the lowest MSE of 993.68 and MAE 890.16. Also, it produces the lowest relative errors percentage (SMAPE) of 23.87% compared to the results of Original and Prediction set.

Figure 6.14 shows visualisation of the spikelets density and segmentation maps predicted using SpikeMulti and SpikeCount and SpikeSEG for one example in the Original set for the Flowering stage of 2016 image series.

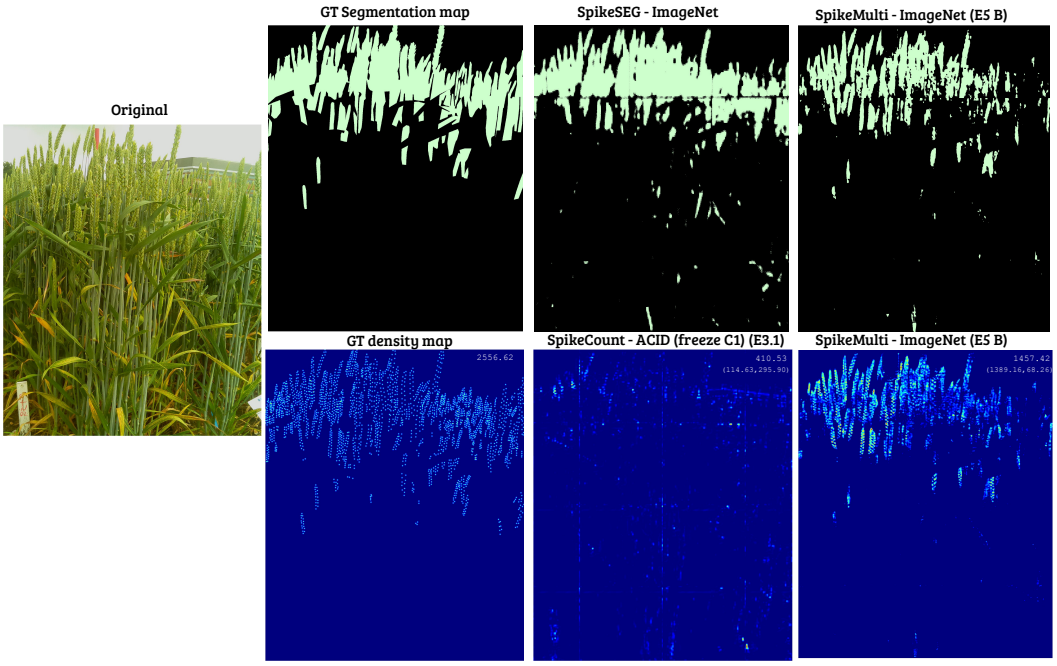

Figure 6.14: Visualisation of the spikelet segmentation and density maps for the **Flowering** stage of 2016 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

Figure 6.15 illustrates that MTL results in the lowest background errors across all experimental setups. MTL and Transfer Learning (E5.2) has minimised the background errors significantly with an MSE of 107.48 and MAE of 105.14. However, in terms of spike region errors, it has led to an increase of 4.63% for MSE and 12.04% for MAE when compared to using Transfer Learning from ACID (E2) for the Original set. Also, it has led to an increase of 29.20% for MSE and 30.46% for MAE when compared to spike region errors from augmentation in the Prediction set.

Table 6.15 shows SpikeMulti performance evaluation when tested on images within the Flowering growth stage in CQ_2017 sequence. Also, Figure 6.17 provides the quantitative errors from the spike and background regions separately.

Table 6.15 shows that MTL has given us the lowest errors across all experimental setups for Original and Prediction sets of the Flowering growth stage. Particularly,

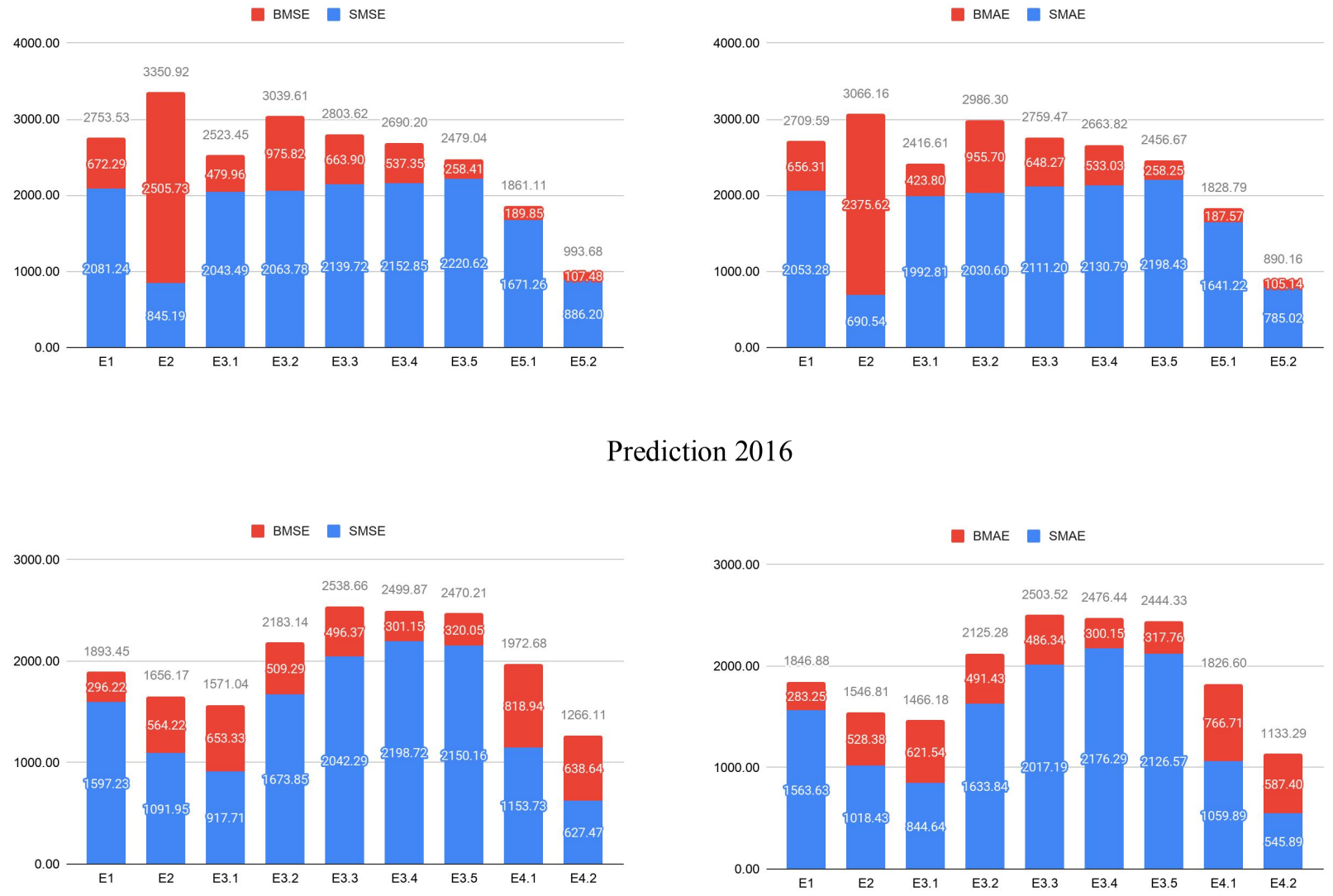

Figure 6.15: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Flowering** growth stage-CQ_2016 sequence

Table 6.15: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Flowering** growth stage of CQ_2017 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	3971.32	3920.14	92.19	2960.75	2915.79	67.57
Loading ACID Weights	5128.09	4679.23	**50.28**	2319.48	2230.08	41.91
Freezing Conv1 Blocks	**3723.46**	**3604.25**	83.77	2140.10	2019.60	35.52
Augmentation-Scratch	-	-	-	1769.35	1555.75	**18.70**
Augmentation-ACID	-	-	-	**1515.68**	**1306.21**	21.08
			SpikeMulti			
Training From scratch	3062.71	3019.54	73.68	-	-	-
Loading ImageNet Weights	**963.53**	**807.21**	**13.73**	-	-	-

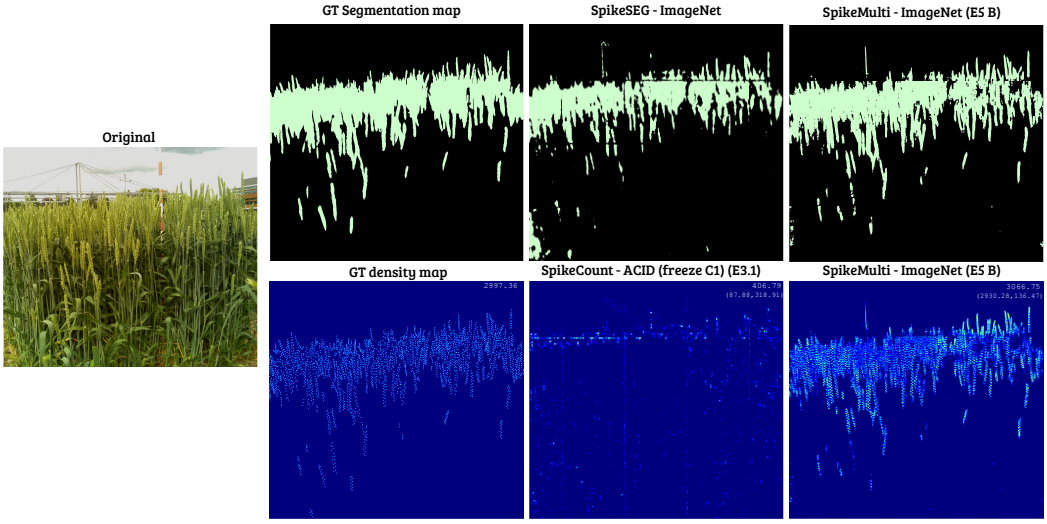

Figure 6.16: Visualisation of the spikelet segmentation and density maps for the **Flowering** stage of 2017 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

utilising both MTL and Transfer Learning has the lowest MSE of 963.53 and MAE 807.21. Also, it produced the lowest relative errors percentage (SMAPE) of 13.73% compared to the results of Original and Prediction set.

Figure 6.17 illustrates that MTL results in the lowest background errors across all experimental setups when applied to the Original set and compared against the Prediction set. MTL and Transfer Learning (E5.2) has minimised the background based errors significantly with an MSE of 121.70 and MAE of 120.25. It also led to

the lowest spike region errors when compared to Original and Prediction sets with MSE of 841.82 and MAE of 686.96.

Figure 6.16 shows visualisation of the spikelets density and segmentation maps predicted using SpikeMulti, SpikeCount and SpikeSEG for one example in the Original set from the Flowering stage of 2017 image series.

6.3.3.4 GS71-73: Grain filling

Lastly, we look at the Grain filling growth stage. As before, we present results of SpikeMulti spikelet estimation for CQ_2016 (Table 6.16) and CQ_2017 (Table 6.17) and report for two sets: Original and Prediction.

Table 6.16: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Grain filling** growth stage of CQ_2016 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	3201.12	3184.25	89.60	2161.81	2136.58	59.69
Loading ACID Weights	3510.72	3348.25	**45.06**	1724.07	1675.39	36.52
Freezing Conv1 Blocks	**2861.43**	**2833.19**	87.08	1513.67	1463.90	29.09
Augmentation-Scratch	-	-	-	2435.74	2376.68	31.57
Augmentation-ACID	-	-	-	**1082.97**	**993.38**	**18.32**
			SpikeMulti			
Training From scratch	2015.90	1998.54	57.24	-	-	-
Loading ImageNet Weight	**650.52**	**588.82**	**12.22**	-	-	-

Table 6.16 shows SpikeCount performance evaluation when tested on images in Grain filling growth stage in the CQ_2016 sequence. In addition, Figure 6.6 provides the quantitative errors from the spike and background regions separately.

Table 6.16 shows that MTL gives the lowest errors across all experimental setups when applied to the Original set and compared against the Prediction set. MTL and Transfer Learning has the lowest MSE of 650.52 and MAE 588.82. Also, it produced the lowest relative errors percentage (SMAPE) of 12.22% compared to the results of Original and Prediction sets.

Figure 6.18 illustrated that MTL gives the lowest background errors across all

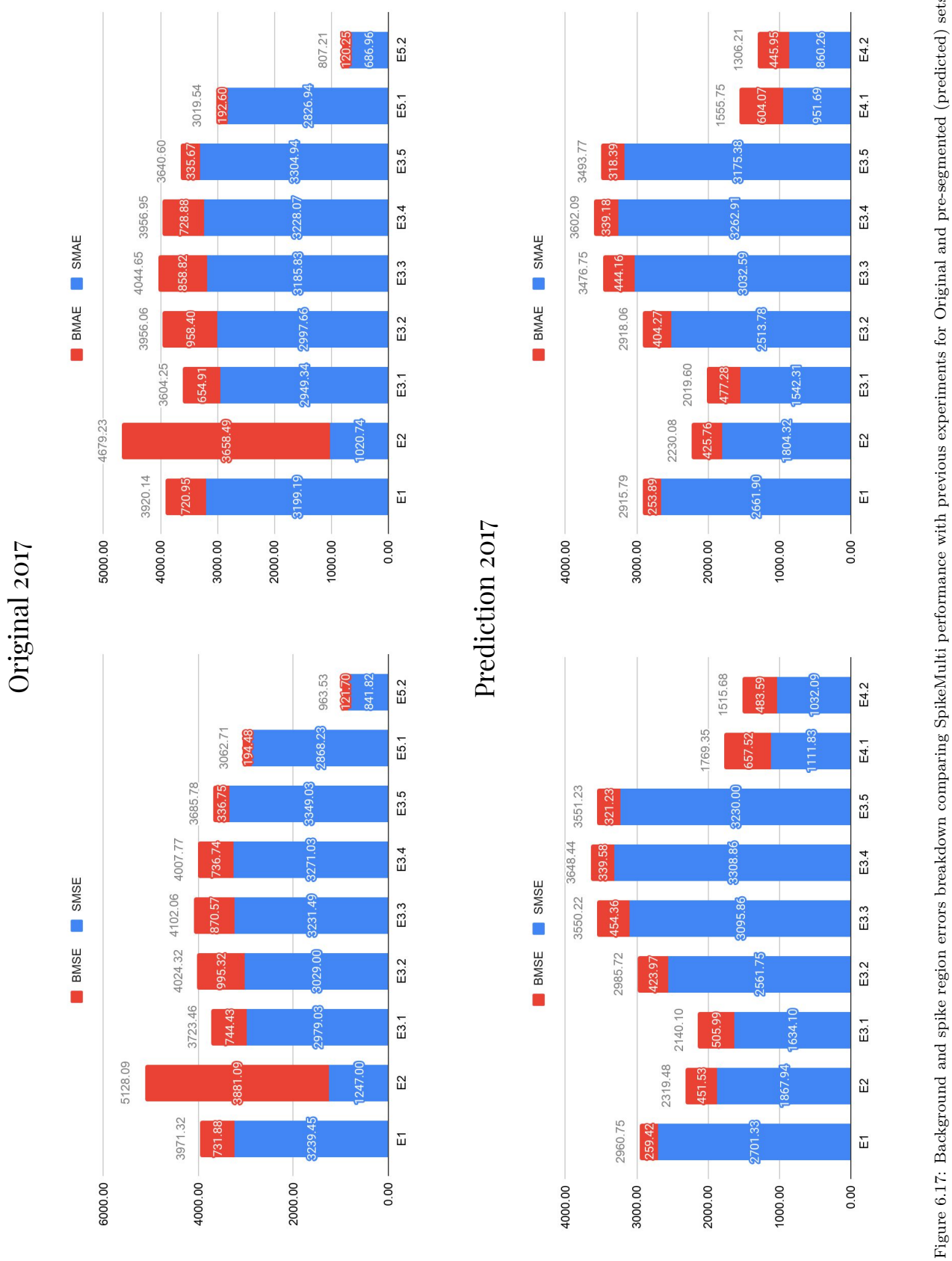

Figure 6.17: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Flowering** growth stage-CQ_2017 sequence

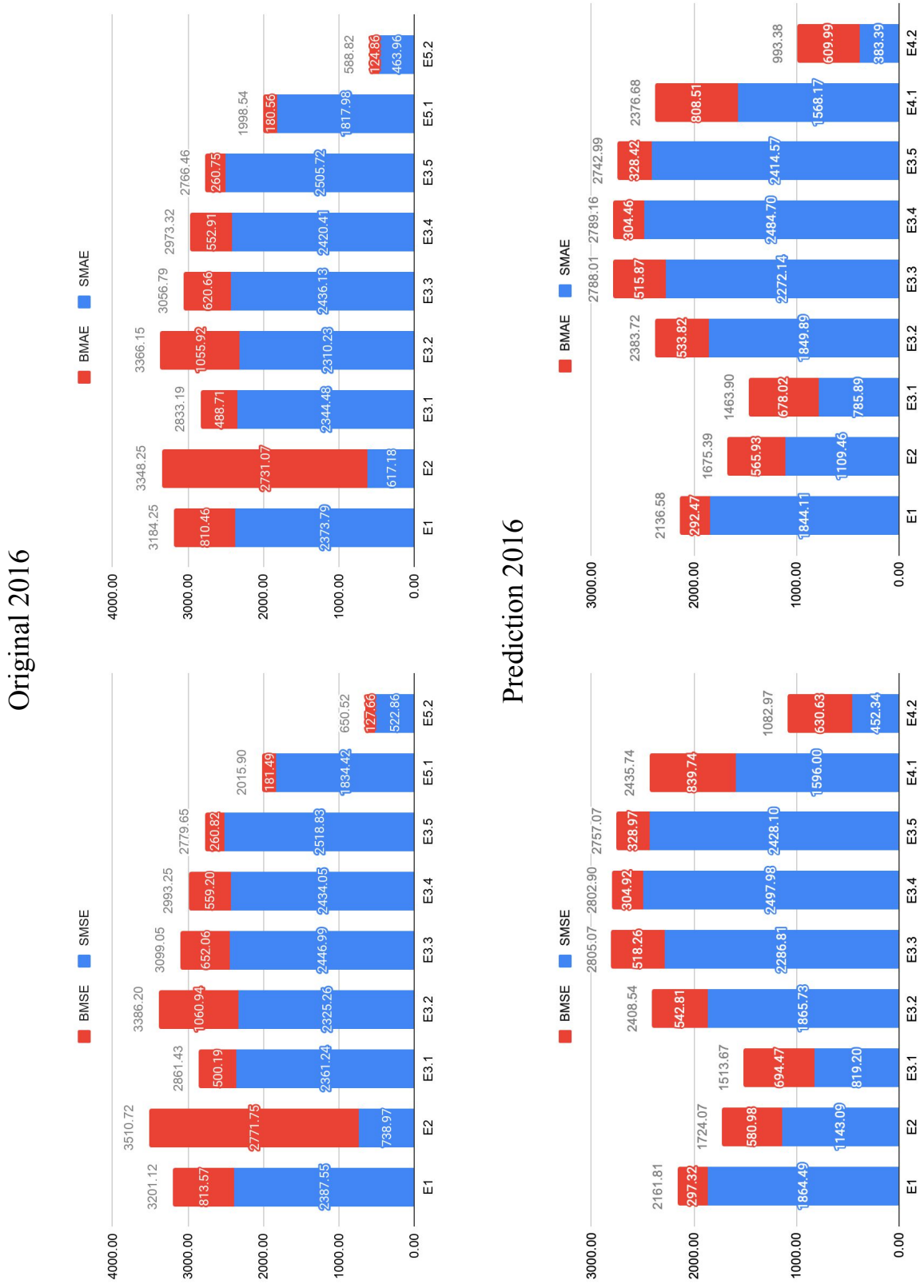

Figure 6.18: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Grain filling** growth stage-CQ_2016 sequence

experimental setups. MTL and transfer (E5.2) has minimised the background based errors significantly with an MSE of 127.66 and MAE of 124.86. It also led to the lowest spike region errors for Original set with MSE of 522.86 and MAE of 463.96. However, it has led to an increase of 13.49% for MSE and 17.36% for MAE when compared to spike region errors resulting from augmentation and Transfer Learning using ACID (E4.2) when testing on the Prediction set.

Figure 6.19 shows visualisation of the spikelet density and segmentation maps predicted using SpikeMulti, SpikeCount and SpikeSEG for one example in Original set from the Flowering stage of 2016 image series.

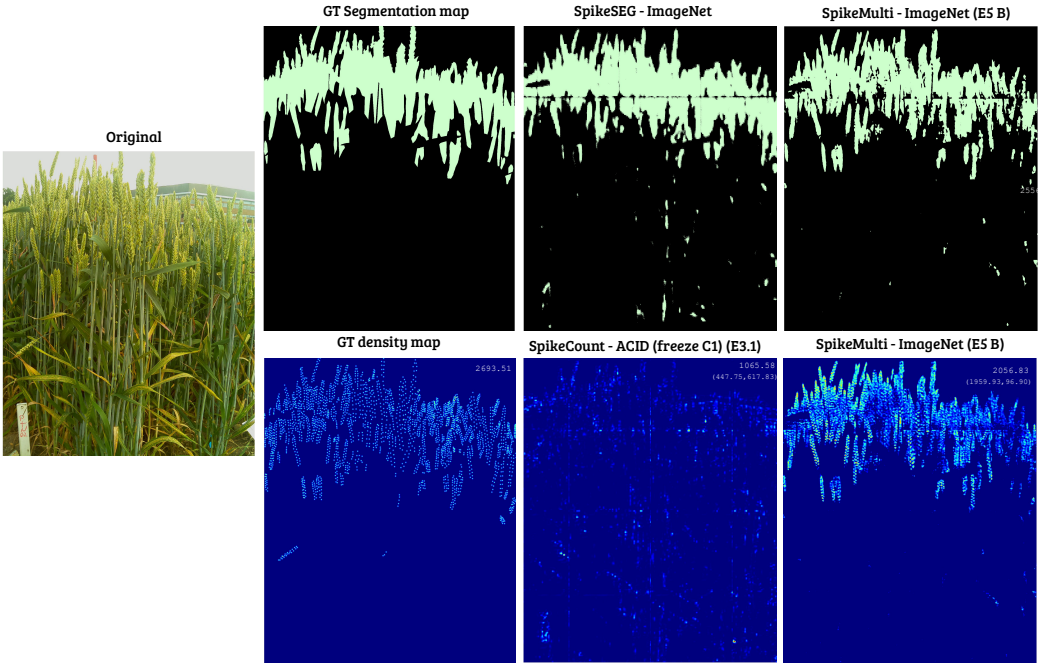

Figure 6.19: Visualisation of the spikelet segmentation and density maps for the **Grain filling** stage of 2016 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

Table 6.17 shows results for the CQ_2017 sequence. Also, Figure 6.20 provides the quantitative errors calculated from the spike and background regions separately.

Table 6.17 shows, as before, that MTL has resulted in the lowest errors across all experimental setups when applied to Original set and compared against Prediction

Table 6.17: Mean Squared Error, The Mean Absolute Error and Symmetric Mean Absolute Percentage Error of estimating the number of spikelets comparing four experimental setups (displayed as rows) and two experimental setups of SpikeMulti of the Original, and pre-segmented by model (predicted) images (displayed in columns) on **Grain filling** growth stage of CQ_2017 sequence

	Original			Prediction		
	MSE	MAE	SMAPE	MSE	MAE	SMAPE
			SpikeCount			
Training From Scratch	4338.18	4298.70	93.22	2988.74	2942.03	66.16
Loading ACID Weights	4848.80	4617.84	**43.56**	2301.58	2208.93	40.87
Freezing Conv1 Blocks	**3758.97**	**3694.02**	82.35	2204.94	2111.12	37.69
Augmentation-Scratch	-	-	-	**1441.42**	**1295.32**	**15.93**
Augmentation-ACID	-	-	-	1511.23	1312.11	20.89.
			SpikeMulti			
Training From scratch	3117.98	3077.92	74.24	-	-	-
Loading ImageNet Weight	**1316.68**	**1263.90**	**21.62**	-	-	-

set. MTL and Transfer Learning (E5.2) has the lowest MSE of 1316.68 and MAE of 1263.90. Also, it produces the lowest relative error percentage (SMAPE) of 21.62%. However, it has increased relative error percentage (SMAPE) by 5.69% when compared to using augmentation (E4.1) for the Prediction set.

Figure 6.20 illustrates same results as for other growth stages. MTL results in the lowest background errors across all experimental setups. MTL and Transfer Learning (E5.2) has minimised the background errors significantly with an MSE of 102.34 and MAE of 101.74. However, in terms of spike region errors, it has led to an increase of 30.59% for MSE and 44.04% for MAE when compared to utilising Transfer Learning from ACID (E2) for Original set. Also, it has led to an increase of 19.93% for MSE and 27.45% for MAE when compared to spike region errors from augmentation in the Prediction set.

Figure 6.21 shows visualisation of the spikelet density and segmentation maps predicted using SpikeMulti, SpikeCount and SpikeSEG for one example in Original set for the Flowering stage of the 2017 image series.

As an overall comparison, Table 6.18 summarise the best experiments done using both SpikeMulti and SpikeCount for every quantitative metrics we used. Employing MTL and ImageNet improved the results of background errors, combined errors and relative error percentages across all growth stages for both CQ_2016 and CQ_2017. For spike region errors, MTL and ImageNet also did best for the Booting stage in

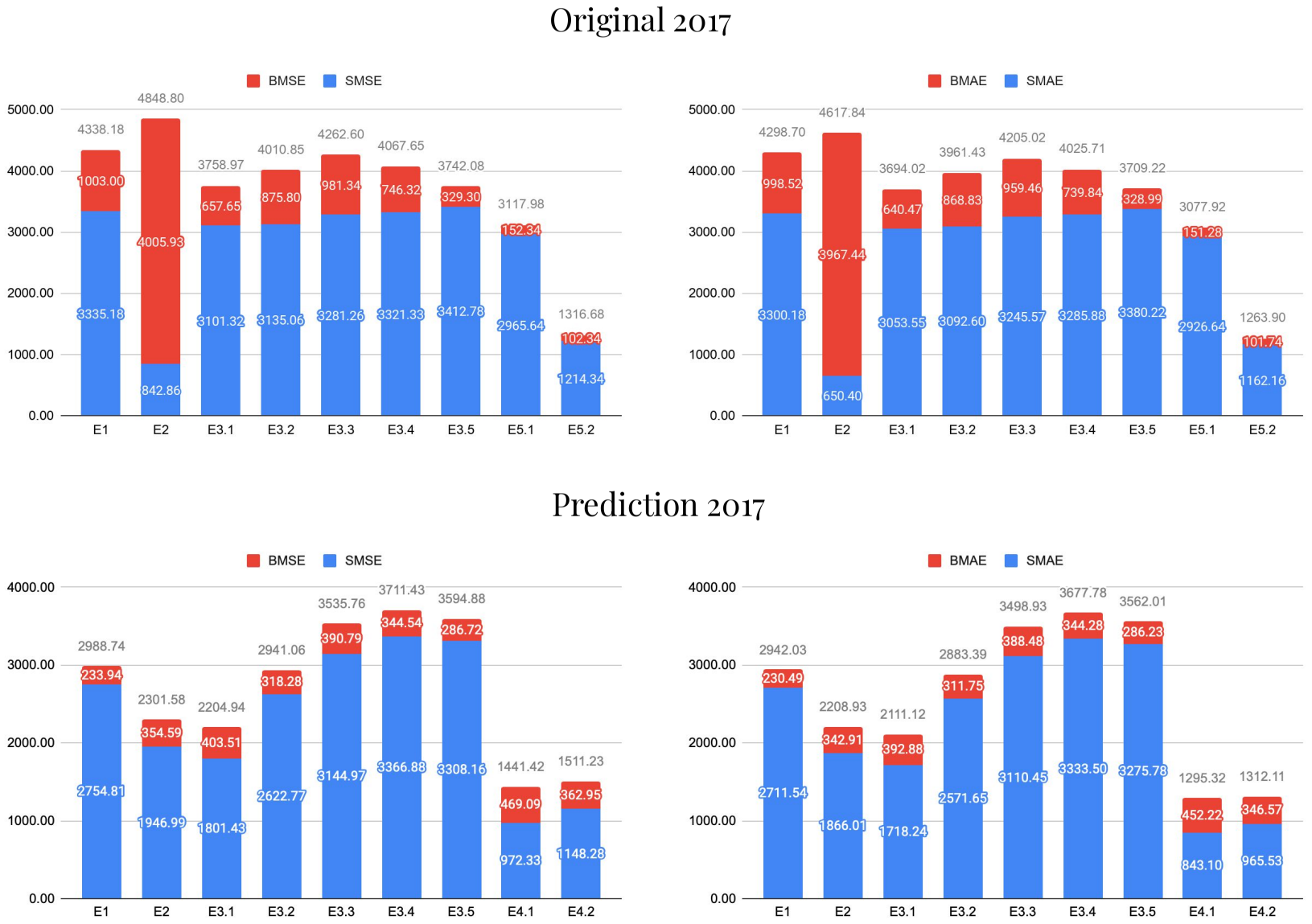

Figure 6.20: Background and spike region errors breakdown comparing SpikeMulti performance with previous experiments for Original and pre-segmented (predicted) sets of **Grain filling** growth stage-CQ_2017 sequence

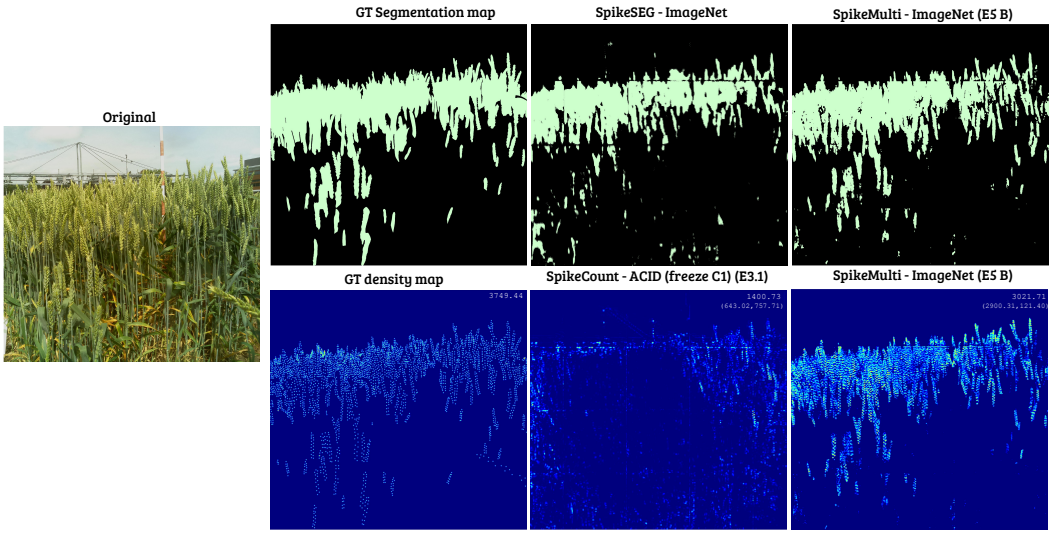

Figure 6.21: Visualisation of the spikelet segmentation and density maps for the Grain Filling stage of 2017 image series. Left most is original image. The first column from top to bottom: (1) Ground Truth (GT) segmentation (2) GT density map with corresponding spikelet number in the scene. Second column: the best segmentation prediction of SpikeSEG (Top) and the best density map of SpikeCount (Bottom) with numbers corresponding to total spikelet numbers predicted in scene (background/spike area). Third column: the segmentation prediction (Top) and density map prediction (Bottom) of SpikeMulti with predicted numbers as before.

Table 6.18: A comparison of best experimental setups for each quantitative metric across Original set for each growth stage for CQ_2016 and CQ_2017

	2016				2017			
Growth stage	Best Background Errors	Best Spike Errors	Best Combined	Best relative error %	Best Background Errors	Best Spike Errors	Best Combined	Best relative error %
Booting	E5 B	E5 B	E5 B	E5 B	E5 B	E5 B	E5 B	E5 B
Heading	E5 B	E2	E5 B	E5 B	E5 B	E2	E5 B	E5 B
Flowering	E5 B	E2	E5 B	E5 B	E5 B	E5 B	E5 B	E5 B
Grain filling	E5 B	E2	E5 B	E5 B	E5 B	E2	E5 B	E5 B

both years and in the Flowering stage in 2017. On the other hand Transfer Learning with SpikeCount did better for spike errors in all but the Booting stage in 2016 and in the Heading and Grain Filling stage in 2017.

6.4 Summary

In this Chapter, we presented a MTL approach to replace the conventional pipeline of applying spike segmentation using SpikeSEG (validated in Chapter 4) and then regressing the spikelet density map using SpikeCount and then inferring spikelet number (validated in Chapter 5). We have trained SpikeMulti, a heterogeneous multitask fully convolutional network, to perform two learning tasks simultaneously (spike segmentation and spikelet density regression) optimising a global cost function which combines cost functions for each task. Then we tested the model on Original set for CQ_2016 and CQ_2017 (with full background) and compared the counting results with SpikeCount performance when tested on Original and Prediction sets. In addition, we compared the segmentation results generated from SpikeMulti to the ones generated from SpikeSEG.

Our findings show that MTL has generated the best combined errors compared to the results of testing SpikeCount on Original and Prediction sets across all growing seasons. This may be because the model is able to differentiate between the background and spikelet regions. This is clear in the CQ_2016 results where, even though there were no improvements in the segmentation results, the model was still capable of identifying the background regions and ignoring them as we saw in the graphs that break down errors into background and spike regions. It is worth noting that segmentation results were improved when testing SpikeMulti when compared to SpikeSEG for some of the growth stages. In CQ_2016, Booting, Heading and Grain filling, Intersection over Union (IoU) metric(s) have increased particularly for early growth stages. For CQ_2017, there are great improvements across all metrics and growth stages. Therefore, it is evident that training spikelet density regression along side segmentation has led to joint features that benefited the segmentation results.

When we compare the results of testing SpikeMulti to SpikeCount on the Original set, SpikeMulti has minimised the combined errors significantly with more than 75%

decrease for both 2016 and 2017 datasets. For the results of testing SpikeCount on the Prediction set, the decrease in combined errors is around 50%. In addition, we noticed that MTL not only leads to a decrease in background errors but also has competitive performance for the spike regions when compared to Prediction set of CQ_2016 and CQ_2017. In addition, it is very clear that analysing the results of background and spike region separately has helped us provide comprehensive analysis particularly when evaluating SpikeMulti performance with SpikeCount performance.

For the various growth stages, our summary Table 6.18 shows that SpikeMulti with ImageNet Transfer Learning has led to the best results across most growing stages, particularly for combined and background errors.

It is worth noting that even though SpikeMulti has robustly managed to reduce the background errors compared to SpikeCount, SpikeCount still outperforms SpikeMulti when tested on the Optimal set for both CQ_2016 and CQ_2017. This might be due to SpikeMulti having led to little or no improvements in spike region errors. This could perhaps be resolved if the ACID dataset was used with MTL in the context of Transfer Learning to improve the counting of spikelets within spike regions.

Chapter 7

Conclusions and further work

Through this project, we have employed fully convolutional model to perform complex tasks (i.e spike segmentation and spikelets density estimation) to analyse a particular yield related measurement, spikelet counting. For this, we used in-field wheat crop images that were composed of three sequences CQ_2015, CQ_2016, and CQ_2017 which represents three growing seasons. We had information on growth stages from each image which enabled us to analyse the difficulty of the estimation in each growth stage. In contrast with many machine learning based indoor phenotypic analysis tasks with ideal lighting and image conditions [121], our work is based on real-world agricultural and breeding situations, where strong wind, heavy rainfall, irrigation and spraying activities can lead to unexpected quality issues.

In order to set up the experiments to achieve the main aim of this thesis, we produced a high quality dataset consisting of three sets of sequences from consecutive years and manually labelled for both segmentation and density estimation tasks, The dataset can be accessed through: `'https://github.com/tanh86/SpikeProject`.

We started by following the workflow of applying segmentation to isolate spike regions and then applying density estimation to extract spikelet number, as this is considered the conventional approach in plant phenomics. In contrast, we also

investigated whether performing MTL for both tasks simultaneously could produce better results.

Another area that we investigated is Transfer Learning. For this we looked at whether employing weights learnt from enormously different data domains in the form of the ImageNet dataset [62] to a different task could be beneficial to extract meaningful traits from wheat crop images. In addition, we investigated using weights learnt from a relatively similar domain, in the form of the ACID dataset trained to perform the same task, in order to improve the model performance.

In section 7.1, we draw insights from the analysis of our results presented in the experimental Chapters (Chapters 4, 5, and 6) and connect to the research questions we posed in Chapter 1. Section 7.2 will preview the open opportunities to further develop the work in the future.

7.1 Discussion

In Chapter 1, we posed three main research questions around the aim of extracting yield traits from wheat crops images captured in uncontrolled settings:

- Can eliminating background from infield wheat images simplify the problem of spikelet counting, given the challenges from the similarity between the background and ROI (spike regions), severe occlusion and high density of spikelets?
- Can Transfer Learning aid in extracting key yield traits in wheat phenotyping when we transfer the knowledge from big dataset containing millions of objects that are drastically different in kind and task from the target dataset? Similarly, can Transfer Learning from more similar wheat images captured indoors under controlled lab settings be beneficial to solve the same task?

- Can MTL in a deep learning setting be beneficial to the problem in hand as opposed to following the conventional workflow that is comprised of a sequence of tasks?

In this section we evaluate how the results of our experiments have enabled us answer those questions.

Overall, through our experiments, we have proved that the deep-learning approach using fully convolutional networks (FCNs) is promising for both spike segmentation and spikelet density estimation problems and the application of Transfer Learning can produce better results for both learning tasks and across the monitored key growth stages.

With respect to our first research question, eliminating background objects and isolation ROI such as spike regions significantly improves extraction yield traits such as spikelet number. Before moving to the impact of isolating the ROI on counting spikelets, we reflect on the results of our spike segmentation proposal, SpikeSEG from 4.

In terms of spike segmentation, our results show that the selection of a larger sub-image sizes for the sliding window in our approach results in better segmentation as it translates to higher classification performance. Our findings are in line with the conclusion reached in the original FCN research [71], where the model trained on original images converged faster than on randomly subsampled patches. In our case, using the whole growth crop images with a size of (2592×1944) was not possible, not only because of computational issues but also because there are many regions without relevant phenotypic information. As a result, we had to use a sub-image to improve the performance of the model as well as to reduce the unnecessary computational complexity. However, SpikeSEG enabled us to select varied sub-image sizes to test performance and computational efficiency during the exploration and we observed that larger images were better. In addition, it is noticeable that enlarging the perception of the model (i.e. selecting larger input

size) was beneficial when learning surrounding objects as it can introduce variation in spike regions such as objects that may appear in sub-images during training. This approach has translated to better segmentation performance for our work.

As verified by experimental results, the model has achieved better outcomes in the later growth stages such as Flowering and Grain Filling. The performance of the SpikeSEG was poor in both Booting and Heading stages and also for spikes partially covered by leaves (Figures 4.4 and 4.5). The main reason behind this, we believe, is that the distribution of images for different growth stages was unbalanced within our data, with limited booting images represented in the training data.

The unique shape of spikes may require more attention around the boundary. In many cases, SpikeSEG was successful to some extent in recovering the spike boundary details, which may be due to fusing the features from three locations in the model (conv3-maxpool, conv4-maxpool, and first upsampling layer). The 2015 training dataset was balanced in terms of different weather conditions, from sunny scenes (high exposure of illumination) to rainy and cloudy scenes. The segmentation of spike regions with high and normal lighting conditions was reasonable. However, the model has captured some background objects that were not present in the training dataset such as grass. For example, Figure 4.5 showed grass regions (to the bottom left of the images) that were wrongly recognised as spike areas. Based on our vision assessment using the method discussed in Chapter 4, this error might be caused by severe light exposure, similar colour- and pattern-based features. Again, we believe that more training data could improve the models to avoid such artefacts.

Moving back to the first question of elimination of background noise, we trained and tested on three CropQuant (CQ) variations sets: Original, Optimal, and Prediction. The Optimal set represents the ideal case where we used the ground truth segmentation mask presented in Chapter 4 to eliminate the noisy background completely from the wheat crop scenes. The Original set represents the opposite case where the model is trained and tested to count spikelets from the images with full

background. Additionally, we implemented a conventional workflow where we apply segmentation and then apply density estimation to predict the number of spikelets. For this, we first extracted the regions predicted from SpikeSEG in Chapter 4 to be spike areas and then tested the SpikeCount model. This case is presented as the Prediction set.

It is clear from Chapter 5 that the best performance results from training and testing SpikeCount on the Optimal set and this is true across all the experiments comparing against the Prediction and Original for both CQ_2016 and CQ_2017. For example, for the Optimal set in both CQ_2016 and CQ2017, the lowest errors were achieved by SpikeCount when using Transfer Learning with the ACID dataset and freezing the first block in the model (conv1 block). Since we have used the ground truth mask to remove the background, we consider these results the baseline for comparison to the other Original and Prediction sets, as those would be more realistic and achievable for real time analysis of crop images.

As we discussed in Chapters 5 and 6, for in-depth analysis of performance, we have broken down the counting errors based on those measured while counting within the spike region and over-counting within the background region for Original and Prediction sets.

After the Optimal set, performance was best on the Prediction set. For this set, performance improved significantly when we used augmentation and Transfer Learning for CQ_2016 and only augmentation for CQ_2017.

For the Original set, which was the most challenging as may be expected, we saw that SpikeCount over-estimates spikelets in the background regions (Figures 5.2 and 5.3), resulting in the worst combined errors. Hence the noise in the background hinders the model's ability to ignore objects in these regions for counting purposes.

We conclude from this that isolating the ROI before extracting important information related to plant development seems to lead to more accurate and robust systems, and this may be the same for similar problems.

With regards to the second research question, from our experiments, it is evident that the application of Transfer Learning can produce better results for spike segmentation and spikelets density estimation and across the monitored key growth stages.

For the first learning task, spike segmentation, we tried Transfer Learning by loading ImageNet parameters produced from a very large set of unrelated images, the ImageNet dataset. This improved the performance of SpikeSEG model for 2016 (Table 4.2) and 2017 (Table 4.3) growing seasons. We postulate that for segmentation, the features obtained from ImageNet through those parameters, even though the objects are dissimilar, still enable us to capture object boundaries accurately resulting in better overall segmentation results.

For the second learning task, spikelets density estimation, we attempted to leverage knowledge from images of wheat captured in a controlled environment. This was done in order to improve extracting key traits from images of the same crop captured in an uncontrolled infield environment. As it was highlighted in Chapter 5, the transferred features from the ACID dataset had a positive effects in general on predicting accurate spikelet number. There are three cases in our experimentation where transferring the knowledge from ACID features has led to best performance. The first case is when initialising the model with ACID parameters has led to the lowest spike region errors in the Original set. The second case is when the features are loaded and then the first conv1 block is suspended. This led to best results in the Optimal set where the background was fully removed. The third case is when used on the Prediction set of the CQ_2016. For this dataset, both augmentation and initialising the model with ACID features has led to best results in regards to spike regions errors (Figure 5.2). The low level block in the VGG16 encoder is associated with the general features such as corners, borders etc. of the objects, in our study those are spikelets. This may explain why freezing this block along with full segmentation has resulted in the best performance. It is clear that general features of spikelets from the ACID dataset lead to robust spikelet identification particu-

larly as ACID images were captured in noise free environments and with consistent lighting which may help with the specific features of the spikelets in wheat. This could imply that plant images captured in indoor environments could help provide more robustness for traits extraction of similar plants captured in more noisy and challenging settings. As a summary, Transfer Learning has a positive effect on both segmentation and counting tasks and potentially could improve the efficacy of any deep learning model in extracting useful key traits from infield uncontrolled crop images. The only case where transfer learning had a negative effect is when we initialised SpikeCount with the ACID weights and continue training the model on the Original set. This is because the images in the Original set were full with the background but they were not present with the same volume in the ACID data. This led to a negative effect of spikelets over-estimating (Figures 5.2 and 5.3) in the background regions.

In relation to the third research question, MTL has shown to be beneficial to our problem. We know that the largest errors from our previous models resulted from SpikeCount when testing on the Original set, where extreme over-counting was present, particularly in background areas. SpikeMulti has significantly minimised those errors for the Original set. For example, the errors were minimised by approximately 75% for CQ_2016 and CQ_2017 (Figures 6.3 and 6.4). Additionally, it is important that we compare the effectiveness of the proposed conventional two step workflow represented by the Prediction set, against the MLT results. We know from Table 6.2 that the MTL model has not improved the segmentation results for CQ_2016. However, looking at figure 6.3 it is clear, from the point of background errors, that MTL has aided in learning joint features from the segmentation task and those have helped with model density estimation as background errors have been minimised compared to the ones measured from the Prediction Set. Even though the Prediction set has lesser background regions than those predicted from the SpikeMulti model given the latter's poorer segmentation performance, SpikeMulti was still able to ignore counting spikelets within the background re-

gions. Hence the combined errors have improved and therefore the overall count has improved by 50% for CQ_2016 and CQ_2017.

It is worth noting that learning joint features from density estimation has a positive impact on improving spike segmentation for CQ_2017, particularly when combined with initialising the MLT model with ImageNet parameters. Additionally, it is noticeable (as reported in Chapter 6) that the Intersection over Union (IoU) metrics have improved notably by 10.47% for MIoU and 16.51% for Spike IoU compared to SpikeSEG segmentation results. This indicates that SpikeMulti has resolved the issue of higher rates of false positive errors, when background region pixels are predicted as spike pixels, which was the downside of SpikeSEG. As a result, it has led to the best performance in relation of background and combined errors. However, for both sequences, it is evident that the ACID-transferred features employed in our SpikeCount model have aided in best ROI (spike regions) counting for both Original and Prediction sets.

It is important to note that SpikeCount when tested on the Optimal set significantly outperformed SpikeMulti for both CQ_2016 and CQ_2017 even though SpikeMulti has robustly managed to reduce the background error compared to SpikeCount. This is likely to be because SpikeMulti has not led to any improvement in spike regions errors. One possible way to resolve this and to make the comparison fairer would be to use ACID knowledge which could improve the counting of spikelets within spike regions. This, however, we leave as further research.

In general, MTL has improved the combined quantitative metrics of counting spikelets and also improved the segmentation which could have potential impact on other applications on extracting important traits from plants.

We now review results for the growth stages for Optimal, Prediction and Original sets. For the Optimal set, freezing the first block in SpikeCount after loading ACID weights has led to the best performance for mature wheat images both in 2016 and 2017 growing seasons. This could be due to the similarity in terms of

spike/spikelets maturity of the ACID dataset and the CropQuant Flowering and Grain filling stages.

We knew that the earlier stages such as Booting and Heading would be harder for counting due to very few developed spikelets in the images. Also there was an additional problem of the unbalanced distribution of images for various growth stages, with limited Booting images represented in the training data, in fact only 5 images, because of the short-term nature of wheat booting as noted in Chapter 3. Furthermore, we also only had 7 images for Heading. This could explain why best results for Booting-CQ_2016 and Heading-CQ_2016-17 were achieved when augmentation was use, as new augmented images may have reinforced the little available training data. For the Booting stage in CQ_2017, Transfer Learning using ACID weights was best.

For the Prediction set, both augmentation variations improved the counting results for all growth stages in both CQ_2016 and CQ_2017 except for Booting in CQ_2016 which was improved using training from scratch.

For the Original set, best models were for most growth stages either either based on simple Transfer Learning using ACID weights or then going on to freeze the first block, conv1. Heading in 2016 and Booting in 2017 benefit from training from scratch.

Lastly, MTL with ImageNet has led to the best combined counting results for all growth stages for both CQ_2016 and CQ_2017. For spike errors only in the Heading, Flowering-CQ_2016 and Grain Filling stages augmentation with both variations within SpikeCount showed some advantage over MTL.

The most important limitation of our research is that even though the models developed in this study have achieved good results in terms of the validation and testing datasets, their generalisation ability will be challenged when we add new unseen images. Thus, it is vital to asses the models' performance and retrain them on more balanced and diverse dataset. This leads us to the second limitation

which is the difficulty of manual labelling. This was an obstacle when this research was conducted as the laboriousness of the process limited the amount of data available for training and testing. Although we managed to annotate the whole dataset we used, manual annotation was extremely time and resource consuming. Lastly, looking at the importance of augmentation in the field of deep learning development, the study was limited in investigating the effect of using augmentation on MTL performance which may have improved the results further.

7.2 Future Work

To improve the results, more images during Booting and Heading, when wheat spikes are emerging, may help performance of CNN-based models. More importantly, images should be as representative as possible, e.g. including different lighting conditions, variety of background objects, and with different image quality. It may be possible to further engage with Transfer Learning by augmenting our image set with images of background objects (e.g. grass areas in different lighting conditions) which could help with better segmentation.

Hence we suggest that, to address in-field phenotypic analysis challenges caused by image quality (a common problem in real-world field experiments), we could use more manually labelled datasets that contain sufficient noise information (e.g. grass and unexpected objects) and ROIs under varied lighting conditions. When possible, comparisons should be performed within similar crop growth stages as those may be more realistic. Another potential solution is to introduce artificially created images to mimic noise and unexpected objects and add them to the training datasets. Another way to generate a more generalised and robust solution is to introduce data augmentations based on deep learning [113] such as adversarial training [135] and generative modelling (GAN) [12]. In earlier growth stages like Booting where there are no spike/spikelet density in the images, another technique that could be explored is counting by detection using Faster-RCNN [96]. Also,

other architectures can be employed such as DeepLapV3+[24], SegNet[8] and Unet [100] which can have potential impact in improving the segmentation results and also can be adapted to perform density estimation.

MTL has shown potential and we could also try to include other main learning tasks such as growth stages classification and auxiliary tasks such as global count regression that could provide improved outcomes. Additionally, employing augmentation such as geometrical transformation, noise injection and random erasing methods [140, 32, 144] could improve MTL performance further.

Part of our future work will also be to deploy the models trained in this work to be part of a more comprehensive plant phenomics system. It may also then be possible to interact with other real-time environmental measurements that will be correlated with key yield traits extracted from the models optimised by this work and other models. Analysis of those measurements and yield traits can then help farmers to better adapt their crops for resilience and to improve quality and would be the ultimate aim to which we have contributed.

This work potentially shows the impact of using Transfer Learning to benefit infield imaging in plant phenomics. Building high-level plant phenomics datasets including those captured in controlled environment or not can be employed to extract traits from other plant in different area. Also, the trained architectures can be fine-tuned to solve similar problems such as spike segmentation and spikelets segmentation on other infield datasets.

Bibliography

[1] N. Alexandratos and J. Bruinsma. World agriculture towards 2030/2050. *Land use policy*, 20(4):275, 2012.

[2] Najmah Alharbi., Ji Zhou., and Wenija Wang. Automatic counting of wheat spikes from wheat growth images. In *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,*, pages 346–355. INSTICC, SciTePress, 2018.

[3] Tahani Alkhudaydi, Daniel Reynolds, Simon Griffiths, Ji Zhou, and Beatriz De La Iglesia. An exploration of deep-learning based phenotypic analysis to detect spike regions in field conditions for UK bread wheat. *Plant Phenomics*, 2019:7368761, 2019.

[4] Tahani Alkhudaydi, Ji Zhou, and Beatriz de la Iglesia. SpikeletFCN: Counting Spikelets from Infield Wheat Crop Images Using Fully Convolutional Networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 3–13. Springer, 2019.

[5] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Interactive object counting. In *European Conference on Computer Vision*, pages 504–518. Springer, 2014.

[6] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. Counting in the wild. In *European Conference on Computer Vision*, pages 483–498. Springer, 2016.

[7] Samuel Arvidsson, Paulino Pérez-Rodríguez, and Bernd Mueller-Roeber. A growth phenotyping pipeline for arabidopsis thaliana integrating image analysis and rosette area modeling for robust quantification of genotype effects. *New Phytologist*, 191(3):895–907, 2011.

[8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[9] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[10] Eric Biot, Millán Cortizo, Jasmine Burguet, Annamaria Kiss, Mohamed Oughou, Aude Maugarny-Calès, Beatriz Gonçalves, Bernard Adroher, Philippe Andrey, Arezki Boudaoud, et al. Multiscale quantification of morphodynamics: Morpholeaf software for 2d shape analysis. *Development*, 143(18):3417–3428, 2016.

[11] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. Crowdnet: A deep convolutional network for dense crowd counting. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 640–644, 2016.

[12] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[13] Rachel Brenchley, Manuel Spannagl, Matthias Pfeifer, Gary LA Barker, Rosalinda D'Amore, Alexandra M Allen, Neil McKenzie, Melissa Kramer, Arnaud Kerhornou, Dan Bolser, et al. Analysis of the bread wheat genome using whole-genome shotgun sequencing. *Nature*, 491(7426):705–710, 2012.

[14] Llorenç Cabrera-Bosquet, José Crossa, Jarislav von Zitzewitz, María Dolors Serret, and José Luis Araus. High-throughput phenotyping and genomic selection: The frontiers of crop breeding converge. *Journal of integrative plant biology*, 54(5):312–320, 2012.

[15] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[16] MP Cendrero-Mateo, O Muller, H Albrecht, A Burkart, S Gatzke, B Janssen, B Keller, N Körber, T Kraska, S Matsubara, et al. Field phenotyping: challenges and opportunities. *Terr. Ecosyst. Res. Infrastruct*, pages 53–80, 2017.

[17] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE, 2008.

[18] Chen Change Loy, Shaogang Gong, and Tao Xiang. From semi-supervised to transfer counting of crowds. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2256–2263, December 2013.

[19] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[20] Scott C Chapman, Torsten Merz, Amy Chan, Paul Jackway, Stefan Hrabar, M Fernanda Dreccer, Edward Holland, Bangyou Zheng, T Jun Ling, and Jose Jimenez-Berni. Pheno-copter: a low-altitude, autonomous remote-sensing

robotic helicopter for high-throughput field-based phenotyping. *Agronomy*, 4(2):279–301, 2014.

[21] Prithvijit Chattopadhyay, Ramakrishna Vedantam, Ramprasaath RS, Dhruv Batra, and Devi Parikh. Counting everyday objects in everyday scenes. *arXiv preprint arXiv:1604.03505*, 2016.

[22] Jinyu Chen and Ao Yang. Intelligent agriculture and its key technologies based on internet of things architecture. *IEEE Access*, 7:77134–77141, 2019.

[23] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *BMVC*, volume 1, page 3, 2012.

[24] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[25] Siu-Yeung Cho, Tommy WS Chow, and Chi-Tat Leung. A neural-based crowd estimation by hybrid global learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(4):535–541, 1999.

[26] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204, 2015.

[27] Joseph Paul Cohen, Genevieve Boucher, Craig A Glastonbury, Henry Z Lo, and Yoshua Bengio. Count-ception: Counting by fully convolutional redundant counting. In *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*, pages 18–26. IEEE, 2017.

[28] Tino Colombi, Norbert Kirchgessner, Chantal Andrée Le Marié, Larry Matthew York, Jonathan P Lynch, and Andreas Hund. Next generation shovelomics: set up a tent and rest. *Plant and Soil*, 388(1-2):1–20, 2015.

[29] Byrd C Curtis, Sanjaya Rajaram, Helena Gómez Macpherson, et al. *Bread wheat: improvement and production.* Food and Agriculture Organization of the United Nations (FAO), 2002.

[30] David Deery, Jose Jimenez-Berni, Hamlyn Jones, Xavier Sirault, and Robert Furbank. Proximal remote sensing buggies and potential applications for field-based phenotyping. *Agronomy*, 4(3):349–379, 2014.

[31] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

[32] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[33] Tao Duan, Bangyou Zheng, Wei Guo, Seishi Ninomiya, Yan Guo, and Scott C Chapman. Comparison of ground cover estimates from experiment plots in cotton, sorghum and sugarcane based on images and ortho-mosaics captured by uav. *Functional Plant Biology*, 44(1):169–183, 2017.

[34] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[35] Luis Manuel Fernández-Ahumada, Jose Ramírez-Faz, Marcos Torres-Romero, and Rafael López-Luque. Proposal for the design of monitoring and operating irrigation networks based on iot, cloud computing and free hardware technologies. *Sensors*, 19(10):2318, 2019.

[36] Jose A. Fernandez-Gallego, Shawn C. Kefauver, Nieves Aparicio Gutiérrez, María Teresa Nieto-Taladriz, and José Luis Araus. Wheat ear counting in-field conditions: high throughput and low-cost approach using rgb images. *Plant Methods*, 14(1):22, Mar 2018.

[37] Luca Fiaschi, Ullrich Koethe, Rahul Nair, and Fred A Hamprecht. Learning to count with regression forest and structured labels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2685–2688. IEEE, 2012.

[38] Fabio Fiorani and Ulrich Schurr. Future scenarios for plant phenotyping. *Annual review of plant biology*, 64:267–291, 2013.

[39] Government Office for Science. The iot: making the most of the second digital revolution. *WordLink.*, pages 1–40, 2014.

[40] Geoffrey French, Mark Fisher, Michal Mackiewicz, and Coby Needle. Convolutional neural networks for counting fish in fisheries surveillance video. *BMVA Press*, 2015.

[41] Geoffrey French, Mark Fisher, Michal Mackiewicz, and Coby Needle. Uea computer vision - image labelling tool. https://bitbucket.org/ueacomputervision/image-labelling-tool, 2015.

[42] Robert T Furbank and Mark Tester. Phenomics–technologies to relieve the phenotyping bottleneck. *Trends in plant science*, 16(12):635–644, 2011.

[43] Mario Valerio Giuffrida, Massimo Minervini, and Sotirios A Tsaftaris. Learning to count leaves in rosette plants. In *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)*, pages 7–10, 2016.

[44] Jason M Green, Heidi Appel, Erin MacNeal Rehrig, Jaturon Harnsomburana, Jia-Fu Chang, Peter Balint-Kurti, and Chi-Ren Shyu. Phenophyte: a flexible affordable method to quantify 2d phenotypes from imagery. *Plant methods*, 8(1):1–12, 2012.

[45] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[46] Wei Guo, Bangyou Zheng, Tao Duan, Tokihiro Fukatsu, Scott Chapman, and Seishi Ninomiya. Easypcc: benchmark datasets and tools for high-throughput measurement of the plant canopy coverage ratio under field conditions. *Sensors*, 17(4):798, 2017.

[47] Mehdi Habibzadeh, Adam Krzyżak, and Thomas Fevens. *White Blood Cell Differential Counts Using Convolutional Neural Networks for Low Resolution Images*, pages 263–274. Springer Berlin Heidelberg, 2013.

[48] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[49] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[50] Md Mehedi Hasan, Joshua P. Chopin, Hamid Laga, and Stanley J. Miklavcic. Detection and analysis of wheat spikes using convolutional neural networks. *Plant Methods*, 14(1):100, Nov 2018.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[52] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[53] Masayuki Hirafuji, Hideo Yoichi, Takuji Kiura, Keiko Matsumoto, Tokihiro Fukatsu, Kei Tanaka, Yukinori Shibuya, Atsushi Itoh, Hirohisa Nesumi, Norihiro Hoshi, et al. Creating high-performance/low-cost ambient sensor cloud system using openfs (open field server) for high-throughput phenotyping. In *SICE Annual Conference 2011*, pages 2090–2092. IEEE, 2011.

[54] Joseph Howse. *OpenCV computer vision with python*. Packt Publishing Ltd, 2013.

[55] Prem Prakash Jayaraman, Ali Yavari, Dimitrios Georgakopoulos, Ahsan Morshed, and Arkady Zaslavsky. Internet of things platform for smart farming: Experiences and lessons learnt. *Sensors*, 16(11):1884, 2016.

[56] Kjeld Jensen, Søren Hundevadt Nielsen, RN Joergensen, A Boegild, NJ Jacobsen, OJ Joergensen, and CL Jaeger-Hansen. A low cost, modular robotics tool carrier for precision agriculture research. In *Proc Int Conf on Precision Agriculture*, 2012.

[57] Anna Kicherer, Katja Herzog, Michael Pflanz, Markus Wieland, Philipp Rüger, Steffen Kecke, Heiner Kuhlmann, and Reinhard Töpfer. An automated field phenotyping pipeline for application in grapevine research. *Sensors*, 15(3):4823–4836, 2015.

[58] Satoshi Kitagawa, Sanae Shimada, and Koji Murai. Effect of ppd-1 on the expression of flowering-time genes in vegetative and reproductive growth stages of wheat. *Genes & genetic systems*, 87(3):161–168, 2012.

[59] Avi C Knecht, Malachy T Campbell, Adam Caprez, David R Swanson, and Harkamal Walia. Image harvest: an open-source platform for high-throughput plant image processing and analysis. *Journal of experimental botany*, 67(11):3587–3599, 2016.

[60] Evgenii Komyshev, Mikhail Genaev, and Dmitry Afonnikov. Evaluation of the seedcounter, a mobile application for grain phenotyping. *Frontiers in plant science*, 7:1990, 2017.

[61] Dan Kong, Douglas Gray, and Hai Tao. A viewpoint invariant approach for crowd counting. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1187–1190. IEEE, 2006.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[63] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1), 2009.

[64] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[65] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[66] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in Neural Information Processing Systems*, pages 1324–1332, 2010.

[67] Yong Li, Zhengyong Cui, Yingli Ni, Mengjing Zheng, Dongqing Yang, Min Jin, Jin Chen, Zhenlin Wang, and Yanping Yin. Plant density effect on grain number and weight of two winter wheat cultivars at different spikelet and grain positions. *PloS one*, 11(5):e0155351, 2016.

[68] Z. Lin and L. S. Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):604–618, April 2010.

[69] Dimitri A Lisin, Marwan A Mattar, Matthew B Blaschko, Erik G Learned-Miller, and Mark C Benfield. Combining local and global image features for object class recognition. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 47–47. IEEE, 2005.

[70] Guillaume Lobet. Image Analysis in Plant Sciences: Publish Then Perish. *Trends in Plant Science*, 2017.

[71] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[72] Hao Lu, Zhiguo Cao, Yang Xiao, Bohan Zhuang, and Chunhua Shen. Tasselnet: counting maize tassels in the wild via local counts regression network. *Plant methods*, 13(1):79, 2017.

[73] Wenhua Ma, Lei Huang, and Changping Liu. Crowd density analysis using co-occurrence texture features. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 170–175. IEEE, 2010.

[74] Bruno Brandoli Machado, Jonatan P M Orue, Mauro S Arruda, Cleidimar V Santos, Diogo S Sarath, Wesley N Goncalves, Gercina G Silva, Hemerson Pistori, Antonia Railda Roel, and Jose F Rodrigues-Jr. Bioleaf: A professional mobile application to measure foliar damage caused by insect herbivory. *Computers and Electronics in Agriculture*, 129:44 – 55, 2016.

[75] Somayya Madakam, Vihar Lake, Vihar Lake, Vihar Lake, et al. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3(05):164, 2015.

[76] Simon Madec, Xiuliang Jin, Hao Lu, Benoit De Solan, Shouyang Liu, Florent Duyme, Emmanuelle Heritier, and Frédéric Baret. Ear density estimation from high resolution rgb imagery using deep learning technique. *Agricultural and Forest Meteorology*, 264:225–234, 2019.

[77] Mark Marsden, Kevin McGuinness, Suzanne Little, and Noel E O'Connor. Fully convolutional crowd counting on highly congested scenes. *arXiv preprint arXiv:1612.00220*, 2016.

[78] Massimo Minervini, Mario V Giuffrida, Pierdomenico Perata, and Sotirios A Tsaftaris. Phenotiki: An open software and hardware platform for affordable and easy image-based phenotyping of rosette-shaped plants. *The Plant Journal*, 90(1):204–216, 2017.

[79] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT press, 1988.

[80] Narendra Narisetti, Michael Henke, Christiane Seiler, Rongli Shi, Astrid Junker, Thomas Altmann, and Evgeny Gladilin. Semi-automated root image analysis (saria). *Scientific reports*, 9(1):1–10, 2019.

[81] Emerson Navarro, Nuno Costa, and António Pereira. A systematic review of iot solutions for smart farming. *Sensors*, 20(15):4231, 2020.

[82] University of Bristol. Wheat: The big picture, 2011. Bristol Wheat Genomics.

[83] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, October 2010.

[84] Siva Kumar Panguluri and Are Ashok Kumar. *Phenotyping for Plant Breeding*. Springer, 2016.

[85] AJD Pask, J Pietragalla, DM Mullan, and MP Reynolds. *Physiological breeding II: a field guide to wheat phenotyping*. CIMMYT, 2012.

[86] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[87] Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3253–3261, 2015.

[88] Dave J. picamera. *Picamera package.* https://picamera.readthedocs.io/en/release-1.13/, 2016.

[89] Michael P Pound, Jonathan A Atkinson, Alexandra J Townsend, Michael H Wilson, Marcus Griffiths, Aaron S Jackson, Adrian Bulat, Georgios Tzimiropoulos, Darren M Wells, Erik H Murchie, et al. Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. *Gigascience*, 6(10):gix083, 2017.

[90] Michael P Pound, Jonathan A Atkinson, Darren M Wells, Tony P Pridmore, and Andrew P French. Deep learning for multi-task plant phenotyping. In *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*, pages 2055–2063. IEEE, 2017.

[91] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[92] V. Rabaud and S. Belongie. Counting crowded moving objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 705–711, June 2006.

[93] Wayne S Rasband et al. Imagej, 1997-2011.

[94] Giulio Reina, Annalisa Milella, Raphaël Rouveure, Michael Nielsen, Rainer Worst, and Morten R Blas. Ambient awareness for agricultural robotic vehicles. *biosystems engineering*, 146:114–132, 2016.

[95] Mengye Ren and Richard S. Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017.

[96] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[97] Daniel Reynolds, Joshua Ball, Alan Bauer, Robert Davey, Simon Griffiths, and Ji Zhou. Cropsight: a scalable and open-source information management system for distributed plant phenotyping and iot-based crop management. *Gigascience*, 8(3):giz009, 2019.

[98] Daniel Reynolds, Frederic Baret, Claude Welcker, Aaron Bostrom, Joshua Ball, Francesco Cellini, Argelia Lorence, Aakash Chawade, Mehdi Khafif, Koji Noshita, et al. What is cost-efficient phenotyping? optimizing costs for different scenarios. *Plant Science*, 282:14–22, 2019.

[99] Matthew Reynolds and Peter Langridge. Physiological breeding. *Current Opinion in Plant Biology*, 31:162–171, 2016.

[100] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[101] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[102] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[103] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[104] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Crowd counting using multiple local features. In *Digital Image Computing: Techniques and Applications, 2009. DICTA'09.*, pages 81–88. IEEE, 2009.

[105] ITU Telecommunication Standardization Sector. Recommendation itu-t y. 2060: Overview of the internet of things. *Series Y: Global informa-*

tion infrastructure, internet protocol aspects and next-generation networks-Frameworks and functional architecture models. Retrieved from https://www. itu. int/rec/T-REC-Y, pages 2060–201206, 2012.

[106] Santi Seguí, Oriol Pujol, and Jordi Vitrià. Learning to count with deep object features. *CoRR*, abs/1505.08082, 2015.

[107] Mikhail A Semenov and Francisco J Doblas-Reyes. Utility of dynamical seasonal forecasts in predicting crop yield. *Climate Research*, 34(1):71–81, 2007.

[108] Ali Shafiekhani, Suhas Kadam, Felix B Fritschi, and Guilherme N DeSouza. Vinobot and vinoculer: Two robotic platforms for high-throughput field phenotyping. *Sensors*, 17(1):214, 2017.

[109] Lindsay M Shaw, Adrian S Turner, Laurence Herry, Simon Griffiths, and David A Laurie. Mutant alleles of photoperiod-1 in wheat (triticum aestivum l.) that confer a late flowering phenotype in long days. *PLoS One*, 8(11):e79459, 2013.

[110] Lindsay M Shaw, Adrian S Turner, and David A Laurie. The impact of photoperiod insensitive ppd-1a mutations on the photoperiod pathway across the three genomes of hexaploid wheat (triticum aestivum). *The Plant Journal*, 71(1):71–84, 2012.

[111] Peter R Shewry. Wheat. *Journal of experimental botany*, 60(6):1537–1553, 2009.

[112] Peter R Shewry and Sandra J Hey. The contribution of wheat to human diet and health. *Food and energy security*, 4(3):178–202, 2015.

[113] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[114] Daniel M Simms, Toby W Waine, John C Taylor, and Graham R Juniper. The application of time-series modis ndvi profiles for the acquisition of crop information across afghanistan. *International journal of remote sensing*, 35(16):6234–6254, 2014.

[115] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[116] Vishwanath A Sindagi and Vishal M Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[117] Vishwanath A Sindagi and Vishal M Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16, 2018.

[118] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[119] François Tardieu, Llorenc Cabrera-Bosquet, Tony Pridmore, and Malcolm Bennett. Plant phenomics, from sensors to knowledge. *Current Biology*, 27(15):R770–R783, 2017.

[120] Apeksha Thorat, Sangeeta Kumari, and Nandakishor D Valakunde. An iot based smart solution for leaf disease detection. In *2017 International Conference on Big Data, IoT and Data Science (BID)*, pages 193–198. IEEE, 2017.

[121] Sotirios A Tsaftaris, Massimo Minervini, and Hanno Scharr. Machine learning for plant phenotyping needs image processing. *Trends in plant science*, 21(12):989–991, 2016.

[122] Jordan Ubbens, Mikolaj Cieslak, Przemyslaw Prusinkiewicz, and Ian Stavness. The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant methods*, 14(1):6, 2018.

[123] Vincent Vadez, Jana Kholová, Grégoire Hummel, Uladzimir Zhokhavets, SK Gupta, and C Tom Hash. Leasyscan: a novel concept combining 3d imaging and lysimetry for high-throughput phenotyping of traits controlling plant water budget. *Journal of Experimental Botany*, 66(18):5581–5593, 2015.

[124] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[125] François Vasseur, Justine Bresson, George Wang, Rebecca Schwab, and Detlef Weigel. Image-based methods for phenotyping growth dynamics and fitness components in arabidopsis thaliana. *Plant methods*, 14(1):63, 2018.

[126] Gabriel Villarrubia, Juan F De Paz, Daniel H Iglesia, and Javier Bajo. Combining multi-agent systems and wireless sensor networks for monitoring crop irrigation. *Sensors*, 17(8):1775, 2017.

[127] Nicolas Virlet, Kasra Sabermanesh, Pouria Sadeghi-Tehran, and Malcolm J Hawkesford. Field scanalyzer: An automated robotic field phenotyping platform for detailed crop monitoring. *Functional Plant Biology*, 44(1):143–153, 2017.

[128] Haohan Wang, Bhiksha Raj, and Eric P Xing. On the origin of deep learning. *CoRR*, abs/1702.07800, 2017.

[129] M. Wang and X. Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *CVPR 2011*, pages 3401–3408, June 2011.

[130] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In C. J. C. Burges, L. Bottou, M. Welling,

Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 809–817. Curran Associates, Inc., 2013.

[131] Yi Wang and Yuexian Zou. Fast visual object counting via example-based density estimation. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3653–3657. IEEE, 2016.

[132] Jeffrey W White, Pedro Andrade-Sanchez, Michael A Gore, Kevin F Bronson, Terry A Coffelt, Matthew M Conley, Kenneth A Feldmann, Andrew N French, John T Heun, Douglas J Hunsaker, et al. Field-based phenomics for plant genetics research. *Field Crops Research*, 133:101–112, 2012.

[133] Ryan Whitford, Delphine Fleury, Jochen C Reif, Melissa Garcia, Takashi Okada, Viktor Korzun, and Peter Langridge. Hybrid breeding in wheat: technologies to improve hybrid wheat seed production. *Journal of experimental botany*, 64(18):5411–5428, 2013.

[134] Bo Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 90–97 Vol. 1, Oct 2005.

[135] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[136] Bolei Xu and Guoping Qiu. Crowd density estimation based on rich features and random projection forest. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

[137] Xiaolin Yang, Seyoung Kim, and Eric P Xing. Heterogeneous multitask learning with joint sparsity constraints. In *Advances in neural information processing systems*, pages 2151–2159, 2009.

[138] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[139] Ian T Young, Jan J Gerbrands, and Lucajs J Van Vliet. Image processing fundamentals, 1998.

[140] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

[141] Jan C Zadoks, Ting T Chang, Cal F Konzak, et al. A decimal code for the growth stages of cereals. *Weed research*, 14(6):415–421, 1974.

[142] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[143] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–841, 2015.

[144] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[145] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.

[146] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[147] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, January 2018.

[148] Ji-chun Zhao, Jun-feng Zhang, Yu Feng, and Jian-xin Guo. The study and application of the iot technology in agriculture. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 2, pages 462–465. IEEE, 2010.

[149] Zhuoyi Zhao, Hongsheng Li, Rui Zhao, and Xiaogang Wang. Crossing-line crowd counting with two-phase deep neural networks. In *European Conference on Computer Vision*, pages 712–726. Springer, 2016.

[150] J Zhou, D Reynolds, T Le Corn, et al. Cropquant: the next-generation automated field phenotyping platform for breeding and digital agriculture. *BioRxiv, Sep*, 2017.

[151] Ji Zhou, Christopher Applegate, Albor Dobon Alonso, Daniel Reynolds, Simon Orford, Michal Mackiewicz, Simon Griffiths, Steven Penfield, and Nick Pullen. Leaf-gp: an open and automated software application for measuring growth phenotypes for arabidopsis and wheat. *Plant methods*, 13(1):1–17, 2017.

[152] Ji Zhou, Daniel Reynolds, . . . , and Simon Griffiths. Cropquant: An automated and scalable field phenotyping platform for crop monitoring and trait measurements to facilitate breeding and digital agriculture. *bioRxiv*, 2017.

[153] Ji Zhou, Francois Tardieu, Tony Pridmore, John Doonan, Daniel Reynolds, Neil Hall, Simon Griffiths, Tao Cheng, Yan Zhu, Dong Jiang, et al. Plant phenomics:: history, present status and challenges. *Journal of Nanjing Agricultural University*, 41(4):580–588, 2018.

[154] Zhang-yue Zhou, Wei-Ming Tian, Ji-Min Wang, Hong-Bo Liu, and Li-Juan Cao. *Food consumption trends in China*. Australian Government Department of Agriculture, Fisheries and Forestry, James Cook University, 2012.

Appendix A

CropQuant Dataset

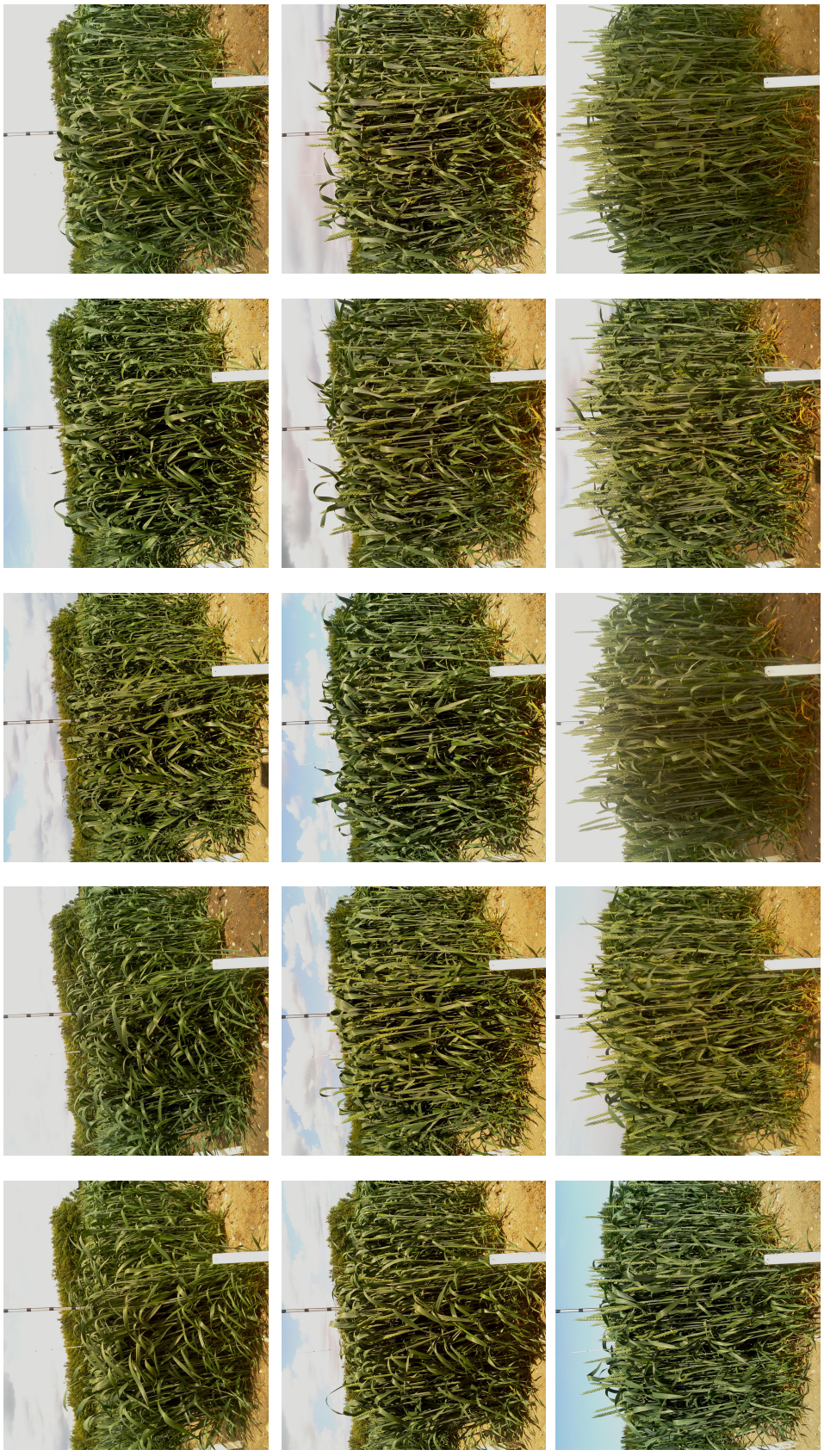

Figure .1: Wheat growth image series in the field collected by CropQuant workstations of the 2015 growing season

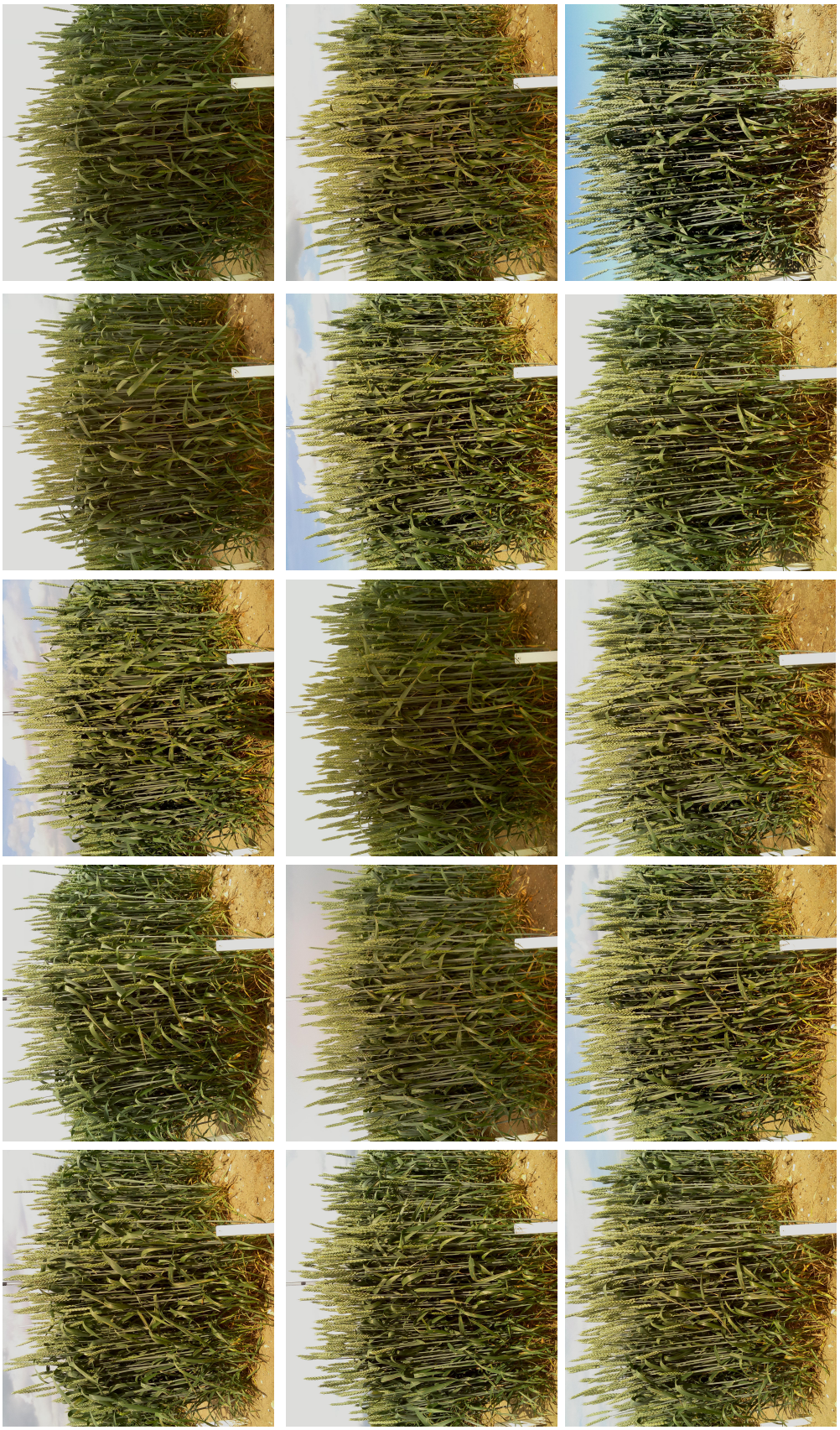

Figure .2: Wheat growth image series in the field collected by CropQuant workstations of the 2015 growing season

Figure .3: Wheat growth image series in the field collected by CropQuant workstations of the 2016 growing season

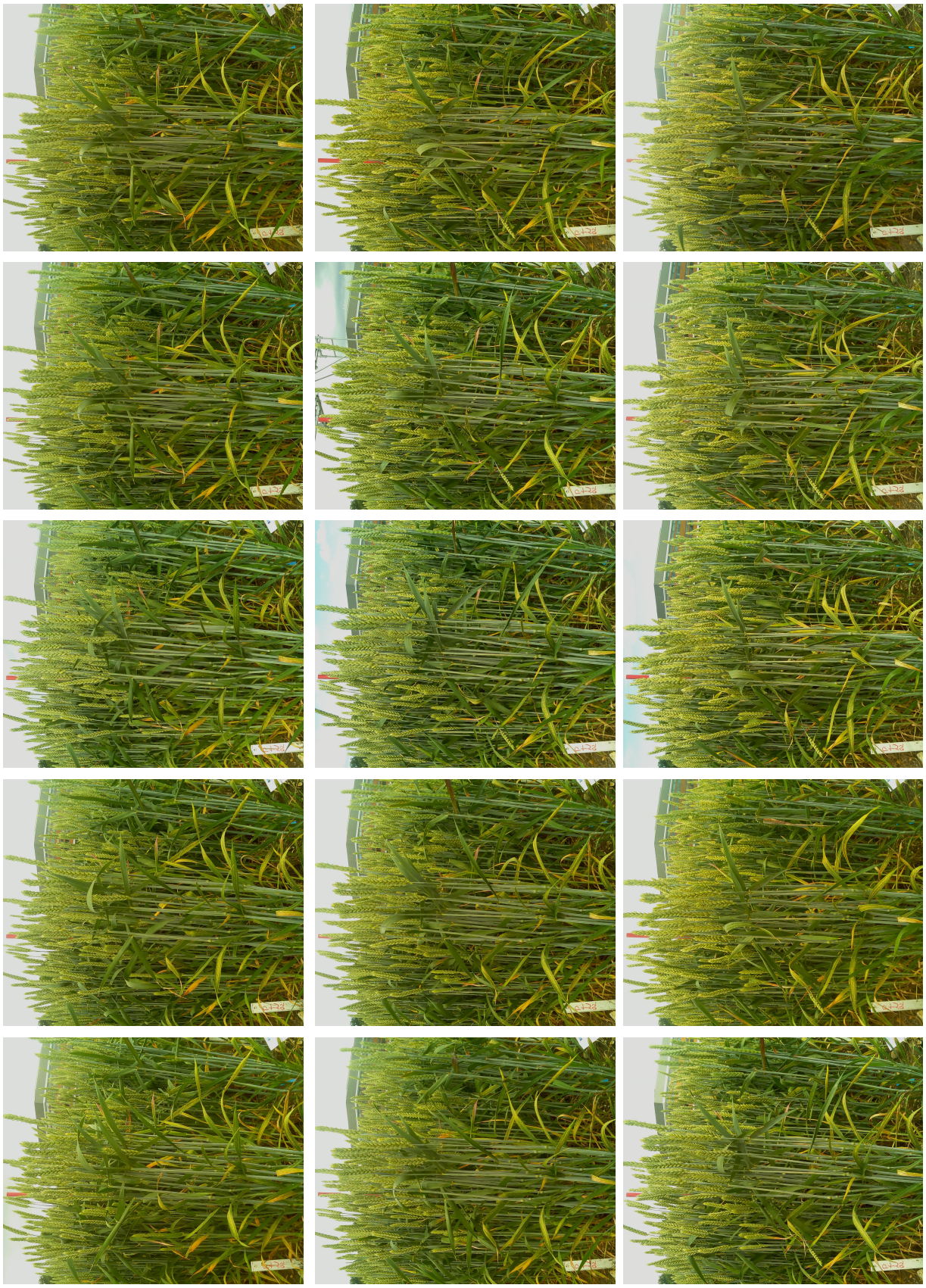

Figure .4: Wheat growth image series in the field collected by CropQuant workstations of the 2016 growing season

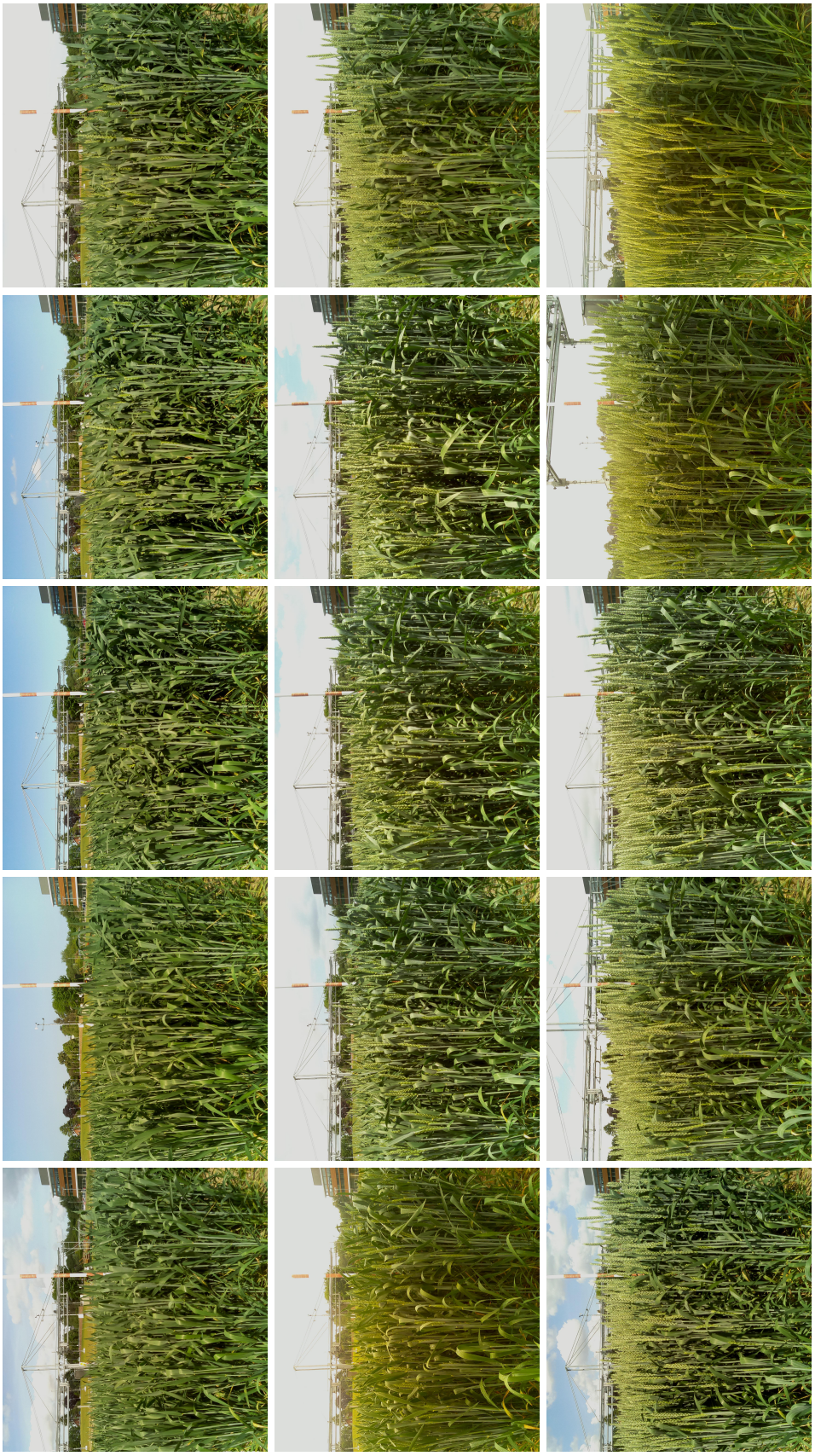

Figure .5: Wheat growth image series in the field collected by CropQuant workstations of the 2017 growing season

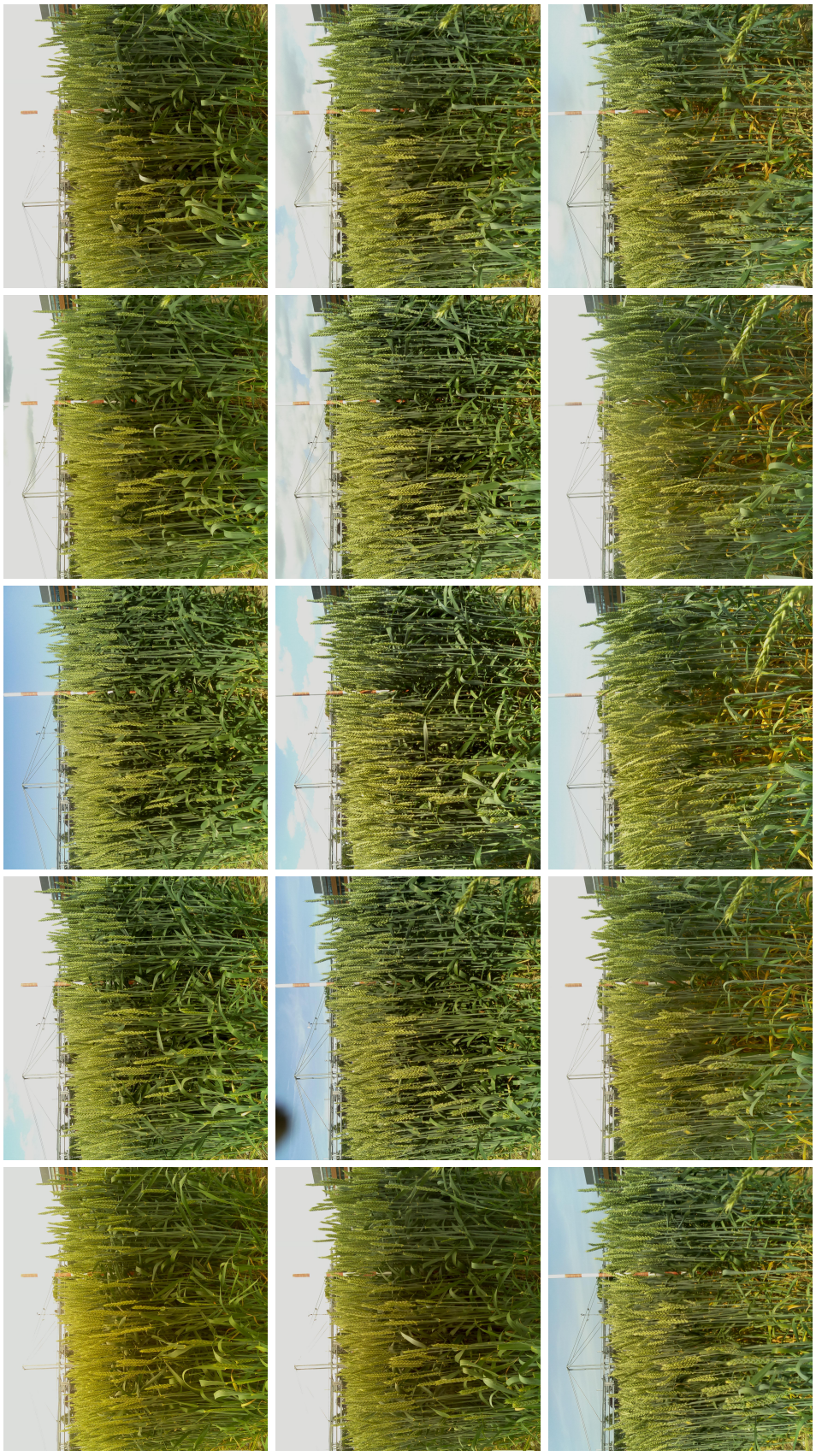

Figure .6: Wheat growth image series in the field collected by CropQuant workstations of the 2017 growing season