Cereal grain and ear detection with convolutional neural networks

Master's thesis
University of Turku
Department of Future Technologies

2020
Joonas Mäkinen

UNIVERSITY OF TURKU
Department of Information Technology
JOONAS MÄKINEN:
Cereal grain and ear detection with convolutional neural networks
Master's thesis, 61 p., 0 app. p.

September 2020

High computing power and data availability have made it possible to combine traditional farming with modern machine learning methods. The profitability and environmental friendliness of agriculture can be improved through automatic data processing. For example, applications related to computer vision are enabling automation of various tasks more and more efficiently.

Computer vision is a field of study which centers on how computers gain understanding from digital images. A subfield of computer vision, called object detection focuses on mathematical techniques to detect, localize, and classify semantic objects in digital images. This thesis studies object detection methods that are based on convolutional neural networks and how they can be applied in precision agriculture to detect cereal grains and ears.

Cultivation of pure-oats poses particular challenges for farmers. The fields need to be inspected regularly to ensure that the crop is not contaminated by other cereals. If the quantity of foreign cereals containing gluten exceeds a certain threshold per kilogram of weight, that crop cannot be used to produce gluten-free products. Detecting foreign grains and ears at the early stages of the growing season ensures the quality of the gluten-free crop.

Keywords: Object detection, neural networks

TURUN YLIOPISTO
Informaatioteknologian laitos
JOONAS MÄKINEN:
Viljan jyvien ja tähkien tunnistaminen konvoluutioneuroverkoilla
Masters, 61 s., 0 liites.

Syyskuu 2020

Suuri laskentateho ja tiedon saatavuus ovat mahdollistaneet modernien koneoppimismenetelmien käytön perinteisen maanviljelyn yhteydessä. Maatalouden kannattavuutta ja ympäristöystävällisyyttä voidaan parantaa automaattisen tietojenkäsittelyn avulla. Yhä useampia tehtäviä voidaan automatisoida tehokkaammin esimerkiksi tietokonenäön avulla.

Tietokonenäkö on tutkimusala, joka tutkii sitä, miten tietokoneet ymmärtävät digitaalisten kuvien sisältämää informaatiota. Hahmontunnistus on yksi tietokonenäön osa-alueista, jossa keskitytään matemaattisiin tekniikoihin, joiden avulla kuvista havaitaan, paikallistetaan ja luokitellaan hahmoja.

Puhdaskauran viljely asettaa viljelijöille erityisiä haasteita. Pellot on tarkistettava säännöllisesti, jolla varmistetaan se, että sato ei ole muiden viljojen saastuttama. Satoa ei voida käyttää gluteenittomien tuotteiden tuottamiseen, jos gluteenia sisältävien viljojen määrä ylittää sallitun rajan painokiloa kohden. Gluteenittoman sadon laatu voidaan varmistaa varhaisessa vaiheessa havaitsemalla vieraiden lajien jyvät ja tähkät.

Asiasanat: Hahmontunistus, neuroverkot

Contents

Chapter 1

Introduction

The continuing expansion of the human population will increase pressure on the agricultural system. Precision farming or sometimes referred to as digital agriculture is a field that uses data-driven applications to enhance agricultural productivity while minimizing its environmental impact. This has created opportunities for machine learning technologies with high-performing computing to unravel possible problem areas of traditional agriculture. [1, p. 1]

In recent years interest in using image processing and computer vision applications in agriculture has grown. One of the factors contributing to this is that the price of the equipment needed for computational power has decreased. Also, the use of automated technologies presents advantages when compared to traditional agriculture. [2, p. 69]

1.1 Problem statement

Inspecting and assessing grain manually is challenging and often a time-consuming process. With machine learning, it is possible to quickly assess large amounts of data. Combining high-performance computing and computer vision techniques have shown potential solving a variety of problems included in traditional agriculture. [2, p. 70]

The goal of this thesis is to study object detection methods and implement a prototype of the detector pipeline based on these findings. The requirements for the

detector are that it must be possible to implement and train on a consumer computer and the quality of the data must be considered when choosing the architecture of the object detection system. Also, the training time must be taken into account when deciding on implementation details.

The detector is trained to identify oats, barley, and wheat from aerial images taken from approximately 4 meters altitude. The goal is to detect individual grains from the oats and whole ears from the barley and wheat. Detection based on these features is natural because each of these cereal plants is identifiable from the traits appearing on grains or ears.

Identifying individual cereals from images opens up opportunities for practical applications. Information can be used in purity control of the grain fields which is essential for the cultivation of gluten-free oats. Foreign grains can contaminate the entire grain crop and it cannot be used in the production of gluten-free products. With the early detection of foreign grains in fields, contamination could be avoided.

The growth density of the cereal plants affects the quantity and quality of the crop. A sufficiently accurate system could be used to estimate plant density and this information could be used to improve the quality of the crop. For example, the amount of water required for high-quality crops varies depending on the density of the plants. This is the one variable that could be assessed with an automated inspection system. It has been stated, that in most growing conditions, nitrogen availability has the greatest impact on cereal quality compared to other nutrients. Also, it does not matter if the nitrogen is released from the soil and crop residues, spread as manures, or applied as artificial fertilizer. Increasing nitrogen availability has positive effects on grain number per unit area. [3, p. 506] Object detection system could be used to estimate grain numbers in certain regions and based on that analysis the nitrogen levels could be optimized.

1.1.1 Gluten-free oat cultivation

People with celiac disease can consume oats as part of gluten-free diet [4] [5]. However, commercial oat products may have been contaminated with grains that contain gluten. Contamination may happen during harvesting, transporting, milling, and processing. The goal of the pure oat production is to produce a gluten-free crop that contains no grains of other cereal or only small quantities. The suitability of a batch of cereal for gluten-free oats is determined by the number of foreign grains in

a batch [6, p. 3-4].

To be classified as gluten-free, cereals must not contain more than 20 milligrams of gluten per kilogram. On average, 5 gluten-containing grains evenly mixed with oats results in 20 milligrams gluten/kilogram oats, which makes the crop unusable in commercial gluten-free products. [7, p. 5].

Poor quality oat crop containing foreign grains causes financial losses to the farmers since if the amount of the foreign grains exceeds permitted limits, the crop cannot be used in the production of gluten-free products.

According to Satafood, purity of the crop can be assessed by collecting random grain sample, going through this sample grain by grain, and count the number of foreign grains [6, p. 6-7]. Detecting foreign cereals by this method requires time-consuming manual labor and this activity could be made more efficient in part by automating inspection-related tasks. If problem areas in fields could be inspected already during the growing season of cereals based on image analysis, financial losses could be avoided.

Careful eradication of gluten-containing cereals from fields is the only effective way to ensure a gluten-free crop. In the first years of gluten-free oat cultivation, weeding the fields from foreign crops must be performed up to three or more times. [7, p. 7]

Fields should be inspected at least twice a year when identifying and uprooting other cereals is as easy as possible. [6, p. 7][7, p. 7]. Automated inspections could be used to estimate workloads in advance and provide information on potential problem areas in the fields. The automated inspection also has the advantage that if it is found that the crop is contaminated with other cereals and it is too laborious to weed the area in question, the cereals can be collected separately from the cereals going to the production of gluten-free products without the need for weeding.

The major risk factor for the cultivation of gluten-free oats is the common wild oat. Sections, where gluten-free oats are produced, must be free of common wild oat because chemical control or weeding it from the oats is impossible and risk areas must be inspected each year at the beginning of the growing season [6, p. 7].

1.2 Related work

In this section studies related to the object detection and problem stated in this thesis are presented. The presented studies show that object detectors based on neural networks can achieve high-performance and accurate results.

1.2.1 Detection of maize tassels from UAV RGB imagery with Faster R-CNN

Liu et al. suggested an object detection system based on Faster R-CNN, residual network (ResNet), and a visual geometry group neural network (VGGNet) for detecting maize tassels from RGB images. The images were collected with an unmanned aerial vehicle with flight height of 15 meters from the ground level. Another dataset was collected using mobile phone camera, 3 meters from ground level. [8, p. 1-7]

Their object detection system reached 94.99% average precision with ResNet101 architecture with UAV images taken from 15 meters from the ground level. VGG16 network achieved 91.51% average precision with the same dataset. [8, p. 8-9]

1.2.2 Using a fully convolutional neural network for detecting locations of weeds in images from cereal fields

In 2018 Dyrmann et al. proposed convolutional object detection system to detect and locate weeds from cereal field images. The goal was to detect mono and dicotyledonous weeds in cereal fields despite possible occlusion. Images were collected from several fields containing different cereals. [9, p. 2-4]

The architecture is based on the SSD512 architecture by Liu et al., (2016). This architecture enables that for locating plants in image it needs only a single forward-pass. Network uses multiple convolution layers to extract features from the input and after the feature extraction the input size is progressively decreased layer by layer to capture multiple scales of the objects. [9, p. 4-5]

Dyrmann et al. reported that the network yielded 0.60 recall and 0.82 precision. Images used in testing phase were relatively similar to those which were used in training phase, resulting that detection accuracy probably will lower when images differ greatly from training data. [9, p. 5-7]

1.2.3 Computer vision-based method for classification of wheat grains using artificial neural network

Sabanci et al. introduced a computer vision system which acquired high accuracy with simplified classification approach. Their goal was to classify Triticum aestivum and Triticum durum wheat grains based on their visual characteristics. Sabanci et al. used RGB images at an angle perpendicular to the grain, which they converted to grayscale, binarized using the Otsu method and segmented using the thresholding operation. The article stated that the visual features each grain is represented by dimension, color and texture. For texture feature extraction they used the gray-level co-occurrence matrix (GLCM). With GLCM contrast, correlation, energy, homogeny, and entropy features were extracted. Artificial neural network with three layers was able to classify wheat grains with 99.9273 percent accuracy and 0.000727 mean absolute error using the whole data set and 21 features. [10, p. 2588-2593]

Chapter 2

Theoretical background

This chapter provides a theoretical overview for machine learning and neural networks which are essential areas of object detection. Some mathematical definitions are covered to explain important concepts related to machine learning. More detailed explanations of mathematics behind machine learning can be found in [11] [12] [13].

Firstly, we will undergo some basic principles of machine learning and briefly discuss a simple linear model and its pitfalls. The linear model, which is presented in this thesis does not directly relate to object detection but it is useful to understand the limitations of this kind of model before discussing more advanced concepts. Secondly, neural network and how they gain insight from the data is discussed. Finally, convolutional neural networks are presented.

2.1 Machine learning

To understand the theory behind neural networks and object detection, we need a brief introduction to the basics of machine learning. Learning done by neural networks is based in part on the same principles as elementary machine learning models. In this section, we will go through some of those relevant main principles which are necessities to gain insight to object detection.

A machine learning algorithm is an algorithm that can learn to perform tasks from the given data. Tasks can vary from classification, regression, machine translation,

etc. Usually, these tasks are hard or impossible to solve with computer programs written by humans. [11, p. 99]

Generally, traditional computer programs are designed to perform computations very fast and follow a list of predefined instructions. For example, if we like to build a system that is able to automatically recognize faces from image, predefining rules for it would be cumbersome. In machine learning, exact rules are not defined but instead, we create a machine learning model that can learn from examples and adjust its parameters to solve the problem. [13, p. 3-4]

2.1.1 Supervised learning

Machine learning and object detection methods considered in this thesis fall into the category of supervised learning. In supervised learning, the training data contains input vectors and the targets. The algorithm will learn the function from this training data, and will output predicted target value for its input vector [6, p. 2-3].

In classification problems, the training data is divided into predefined categories which represent the correct target values for corresponding input vectors [12, p. 2-3]. Classification algorithms have a crucial part in machine learning and object detection. The learning algorithm produces a function which can classify inputs in k categories.

$$f : R^n \rightarrow \{1, ...k\} \tag{2.1}$$

Model can be described as function where x is the input vector and output y is the predicted category [11, p. 100].

$$y = f(x) \tag{2.2}$$

2.1.2 Generalization

Generalization can be defined as a machine learning model's ability to perform correctly with the data that differs from the training data. Traditionally, the training data can hold only a small fragment of all possible input vectors. Consequently, the model must be able to produce the exact form of the function y(x) based on the learning phase. [12, p. 2]

Effectively, finding the best fitting function means that the model is minimizing the training error [11, p. 113]. Training error can be measured by running the model with training data. Generalization error or test error is the expected value of the

error on a new input [11, p. 110]. Overfitting will be effected if the gap between training error and test error grows too large. The model is underfitting when it cannot produce low training error. Inhibiting overfitting and underfitting, the model can minimize the gap between training and test error and making training error small. [11, p. 111]

2.1.3 Linear model

The perceptron algorithm is an example of an algorithm which produces a linear discriminant model, which can be used to solve linearly solvable two-class classification problem [12, p. 192].

$$f(x) = \begin{cases} 1 \; if \; w \cdot x + b > 0, \\ 0 \; otherwise \end{cases} \tag{2.3}$$

Where w is a vector of weights, x is the input, and b is the bias which shifts the decision boundary. The output of the function is 0 or 1, which represents the classes. [14]

By calculating error function and applying stochastic gradient descent, while iterating through training data, we have created a simple machine learning model. If the training data is linearly separable, the algorithm will find a solution to the binary class problems. However, the perceptron will not be able to solve the problem, if the data is not linearly separable. [12, p. 192-195]

A phenomenon called the curse of dimensionality is arising in many fields in computer science, and especially in machine learning. This phenomenon occurs when the number of dimensions in data is high. Machine learning problems become complex and the number of possible configurations of a set of variables increases exponentially as the number of variables increases. [11, p. 155-156]

With high dimensional data, there might not be enough training data in some regions in the feature space [15, p. 55]. Simple machine learning models that form linear combinations of fixed basis functions are limited by the curse of dimensionality [12, p. 225]. Neural networks address this problem by adaptive basis functions and adjustable parameter values which are learned during training [12, p. 226].

2.2 Neural networks

The neural network consists of layers and neurons, which are connected to each other. This structure is loosely based on neuroscience. Neural networks form the basis for object detection and many other real-world applications. [11, p. 168] The computing power of a neural network is derived from the parallel distributed structure and ability to learn from data [16, p. 2]. Neurons in the network can adapt their weights to changes in the surrounding environment. This forms the basis for retraining the network in different environments. [16, p. 3]

2.2.1 Neuron

Neuron is a unit, which receives information as inputs from other neurons, processes this information and finally sends it forward to other neurons. Input is in form of $x = [x_1, x_2...x_n]$ which are multiplied by some weights $w = [w_1, w_2...w_n]$. After receiving input, neuron processes the information by summing the dot product of the inputs and weights. In some cases constant value called bias is added to dot product. [13, p. 7-8]

$$z = \sum_{i=0}^{n} x_i w_i \tag{2.4}$$

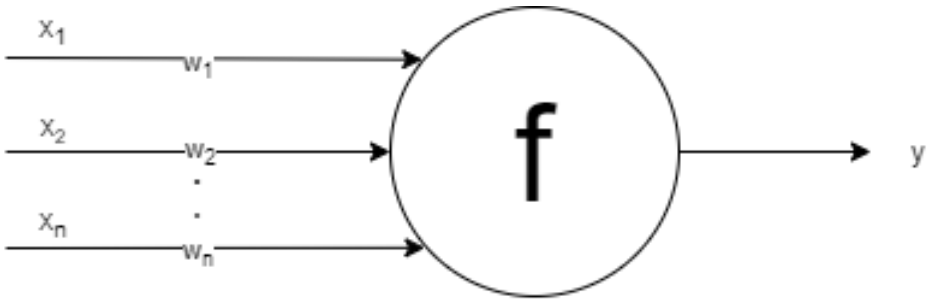


Figure 2.1: Illustration of neuron.

This sum is then processed by neuron's activation function which produces the final output and which can be passed to other neurons [13, p. 7-8].

$$y = f(x \cdot w + b) \tag{2.5}$$

2.2.2 Activation

Previously, perceptron and its ability to output values 0 or 1 were discussed. Slight changes in weights or bias values might flip perceptrons output from 0 to 1 or vice versa. This behavior can affect the whole network and altering weights and biases can become complicated. [14] However, the structure of the data and nature of the problem will affect the choice of the activation function [12, p. 227]. This section covers the three most commonly used [13, p. 13] activation functions.

Sigmoid function outputs a value between 0 and 1, and when the input is large the output is close to 1, or if the input is small the output will be close to 0. The output from this neuron is always a positive value. [13, p. 13-14] With extreme values sigmoid saturates, meaning that output becomes flat [11, p. 68]. Saturation can shrink the gradients, which will affect the learning process in a negative way [11, p. 184].

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.6}$$

A similar kind of S-shaped nonlinearity occurs on neurons using hyperbolic tangent (tanh) activation function. However, tanh neurons output ranges from -1 to 1 [13, p. 13-14].

The shape of the rectified linear unit (ReLU) differs from sigmoid and tanh, resembling hockey-stick shape using function f(z) = max(0,z). ReLU is widely used in tasks related to computer vision [13, p. 13-14]. The ReLU is a broadly reminiscent of a linear unit but differs from it by outputting zero values half of its domain. With active ReLU derivatives are large which makes gradients consistent and large [11, p. 193]. According to Krizhevsky et al. ReLUs carry the property that they do not require input normalization to prevent them from saturating. ReLU requires only some training examples to produce positive input, and that unit will learn. [17, p. 4]

When the network is trained with gradient descent, saturating nonlinearities are slower than non-saturating nonlinearity [17, p. 3]. Krizhevsky et al. reported that deep convolutional neural network with ReLUs reached 25 percent training error six times faster than the network with tanh units on CIFAR-10 dataset [17, p. 3].

2.2.3 Layers

In feedforward networks the data is fed through the layers of network, while it learns parameters which will result function approximation. Combining the layers allows network to chain functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. In this example the $f^{(1)}$ represents the first layer and $f^{(2)}$ represents the second layer. [11, p. 168]

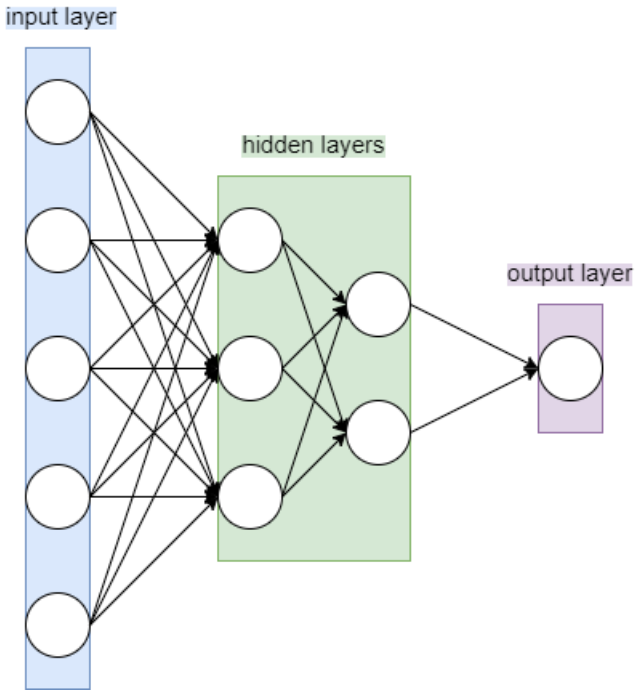


Figure 2.2: Illustration of network layers.

The layer on the left can be referred as input layer, which contains input neurons. Middle layers contain hidden neurons, and they are not used as inputs or outputs of the network. This layer is usually referred as a hidden layer. The output layer contains the output neurons of the network. The number of neurons can vary on each layer, depending on the data and the problem the network is trying to solve. [14]

In feedforward network, the number of input neurons is the same as measurements in data set [18, p. 26]. For example, each pixel in a picture can be seen as one measurement. The number of neurons in the output layer is determined by the number of output classes. Each neuron representing one of the possible classes. [18, p. 26]

A multi-layer feedforward network has three characteristics that make learning from

training data possible. Firstly, each neuron has a nonlinear activation function. Nonlinearity in input-ouput relation is important because that is what sets it apart from the single-layer network. Secondly, the network contains at least one hidden layer. Hidden layers enable learning more complex features from data. Lastly, the network has a high degree of connectivity between neurons. [16, p. 157]

2.2.4 Loss function

The loss function measures the fitness of the model by quantifying the distance between the real and predicted value of the target. The function's output is called loss and it is usually a non-negative number. The more accurate predictions are, the closer the loss is to 0. [19, p. 91]

The loss can be calculated in many ways however one of the most used loss functions in regression problems is the sum of squared errors. For prediction $\hat{y}^{(i)}$ when true label is $y^{(i)}$, squared error is defined by:

$$l^{(i)}(w,b) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2 \tag{2.7}$$

To obtain loss over entire dataset:

$$L(w,b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(w,b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(w^T x^{(i)} + b - y^{(i)})^2 \tag{2.8}$$

Ultimately, the goal is to find parameters (w*, b*) that minimize the total loss over entire training data.

$$w^*, b^* = argmin_{w,b} L(w,b) \tag{2.9}$$

Another loss function for regression is Smooth L1 loss, which is less sensitive to outliers and may prevent exploding gradients in some cases. Smooth L1 loss avoids exploding gradient problem by calculating squares only for values less than 1.

$$L(x,y) = \begin{cases} \frac{1}{2}(x-y)^2 \ \ if \ |x-y| < 1, \\ |x-y| - \frac{1}{2} \ \ otherwise \end{cases} \tag{2.10}$$

For classification problems, we can measure performance with cross-entropy loss. The model outputs a probability value between 0 and 1. The loss value increases as

the predicted value diverge from the actual value. The cross-entropy loss function penalizes those decisions more if the confidence of the prediction is high but the predicted label is wrong. For binary classification problems the loss function can be defined as the following:

$$L(y, p) = -(y\ log(p) + (1 - y)\ log(1 - p)) \tag{2.11}$$

Cross-entropy can also be used for multi-class problems. In these problems, we calculate a separate loss for each class label per observation and sum the result.

$$L(y, p) = -\sum_{c=1}^{M} y_{o,c}\ log(p_{o,c}) \tag{2.12}$$

Where M is the number of classes, log is the natural log, y binary indicator if the label c is the correct classification for observation o, and p is the predicted probability.

2.2.5 Gradient descent

In many cases, finding parameters that will land on functions global minimum is not an easy task. With non-convex functions with multiple local minimums, simple analytical solutions cannot be applied. However, when the loss surfaces are non-convex and high-dimensional gradient descent can be applied in the training phase to find some local minimum. Gradient descent updates parameters iteratively in the direction that incrementally lowers the loss function. There is no guarantee that gradient descent will converge to a global minimum but it will descent towards a local minimum. [19, p. 92]

Gradient descent is a rather naive approach to update parameters since it consists of taking the derivative of the true loss. In other words, it computes the average of losses on entire training data and updates parameters based on that calculation. Traditionally, this is computationally inefficient and slow. [19, p. 92] Stochastic gradient descent (SGD) uses a small proportion of training data called minibatch β on each iteration. The derivative of average loss is calculated based on minibatch instead of the entire training data and parameters are adjusted accordingly. This produces the gradient which is multiplied with some predetermined step size η. Finally, this multiplication is subtracted from current parameter values to obtain updated values. [19, p. 92-93]

$$(w, b) \leftarrow (w, b) - \frac{\eta}{|\beta|} \sum_{i \in \beta} \partial_{(w,b)} l^{(i)}(w, b) \tag{2.13}$$

SGD cannot achieve local minima in finite number of steps but it will slowly converge towards it. This means that the training phase must be stopped when a predetermined number of iterations is reached or some other stopping criteria is fulfilled. [19, p. 93]

Adaptive Moment Estimation (Adam) is an extension of SGD. The idea behind Adam is to compute adaptive learning rates for each parameter. First, exponentially weighted average of past gradient are computed (v_{dW}). Second, the exponentially weighted average of the squares of past gradients are computed (s_{dW}). Third, bias correction is applied to computed averages ($v_{dW}^{corrected}, s_{dW}^{corrected}$). Lastly, the parameters are adjusted based on corrected averages.

$$\begin{aligned} v_{dW} &= \beta_1 v_{dW} + (1 - \beta_1)\frac{\partial J}{\partial W} \\ s_{dW} &= \beta_2 s_{dW} + (1 - \beta_2)(\frac{\partial J}{\partial W})^2 \\ v_{dW}^{corrected} &= \frac{v_{dW}}{1 - (\beta_1)^t} \\ s_{dW}^{corrected} &= \frac{s_{dW}}{1 - (\beta_1)^t} \\ W &= W - \alpha \frac{v_{dW}^{corrected}}{\sqrt{s_{dW}^{corrected}} + \epsilon} \end{aligned} \tag{2.14}$$

Where v_{dW} is the exponentially weighted average of past gradients, s_{dW} is the exponentially weighted average of past squares of gradients, β_1 and β_2 are tunable hyperparameters, $\frac{\partial J}{\partial W}$ is the cost gradient with respect to current layer, W is the weight matrix, α is the learning rate and ϵ is a small value to avoid dividing by zero. [20]

2.2.6 Backpropagation

In feedforward networks, input x flows through network producing some estimation of y as output. This operation is known as forward propagation. After the input vector is propagated forward through network, the estimate of y is compared then to the true value of y. This comparison is done by using loss function. [11, p. 204]

The backpropagation algorithm computes the gradient of loss function by using the chain rule of calculus. This technique allows the algorithm to adjust weight and bias values in a manner that the total error is minimizing. [11, p. 204-207] In other words, backpropagation calculates how much changing the weights and biases in the network will affect the loss function. This can be achieved by calculating partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$ [14].

2.2.7 Regularization

A central problem in training a machine learning model is how to train such a model that performs well on new inputs that are not included in the training data. There are strategies that are designed to reduce testing error costing increased training error. These strategies are referred as regularization. [11, p. 228] Regularization can be seen as a modification of the objective function of the model. [13, p. 35]

With the dropout regularization strategy, overfitting can be prevented. Dropout keeps neuron active with certain probability during the training phase or sets it to zero. This prevents network to become dependent on certain neuron or combinations of neurons since it has to perform accurately even when some parts of the network are dropped out. [13, p. 36]

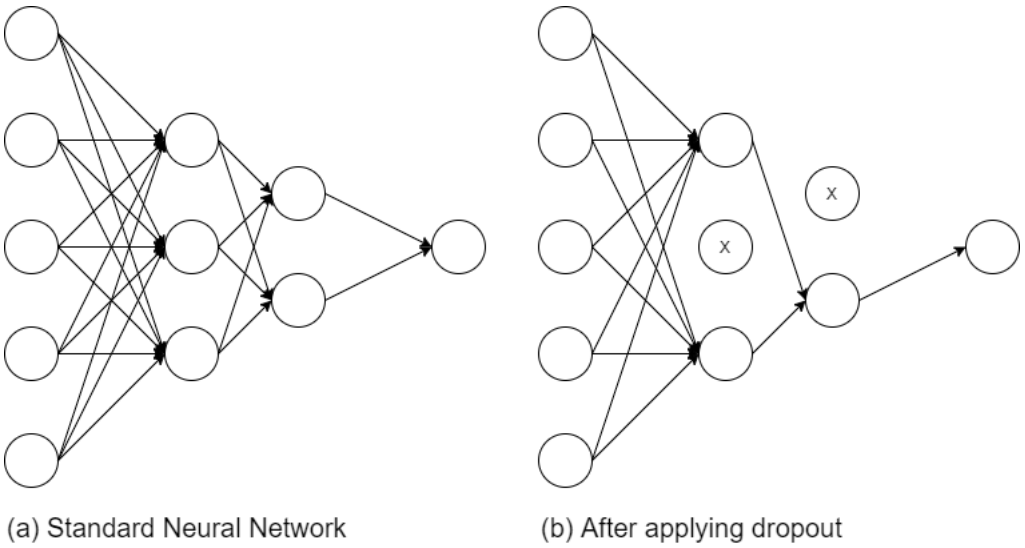


Figure 2.3: Illustration of network before and after dropout has been applied.

Combining models improves the performance of machine learning methods, but with large networks that require long training times this strategy becomes too expensive. Training different architectures and finding optimal hyperparameters for

each architecture is a cumbersome task. Dropout provides a way to combine different network architectures efficiently. [21, p. 1930]

Srivastava et al. reported that training the network with dropout may take 2-3 times longer compared to the model without dropout. This creates a trade-off between training and training time since we are training a slightly different architecture of the model at every training batch when using dropout. [21, p. 1952]

2.3 Convolutional neural networks

The feature selection process in machine learning is cumbersome and limiting factor, especially in computer vision related tasks. In fully connected networks 28x28 pixels input would add up to 784 incoming weights in the hidden layer and 200x200 pixels input would add up to 120 000 weights. This technique runs into problems when image size grows. [13, p. 89]

Convolutional neural network (CNN) finds solution to this problem by reducing parameters and connections in adjacent layers. In the convolutional layer, neurons are connected to a small local region of the previous layer. [13, p. 89] Arranging neurons this way ensures that the network is extracting local features that are located in small subregions of the input. Usually, pixels in an image that are close to each other, are strongly correlated and these local features can be combined and later used to detect higher-order features in the image. [12, p. 267-268] CNNs have thrived in computer vision applications such as object detection, face recognition, robotics, and self-driving cars [22, p. 2].

2.3.1 Convolution layer

The most common architecture of a convolutional neural network consists of convolutional layers, pooling layers, and fully connected layers. The convolutional layer utilizes kernels to process the image with convolutional function which produces a feature map as output. [22, p. 2-3], [11, p. 331-332] This function takes two arguments first being the input and second the kernel. In the machine learning context, the input is usually a multidimensional array of data while the kernel is a multidimensional array of parameters. [11, p. 331-332] For a two-dimensional image I and two-dimensional kernel K, convolution $(K * I)(i, j)$ can be expressed as

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.15}$$

In many cases, machine learning libraries implement cross-correlation and call it convolution. This is done because usually cross-correlation is easier to implement traditionally and the effect on network performance is the same. [11, p. 332-333] In the convolutional layer, the cross-correlation is calculated between input and kernel to produce an output. Typically, in the training phase, the kernels are initialized to random values. [19, p. 234]

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.16}$$

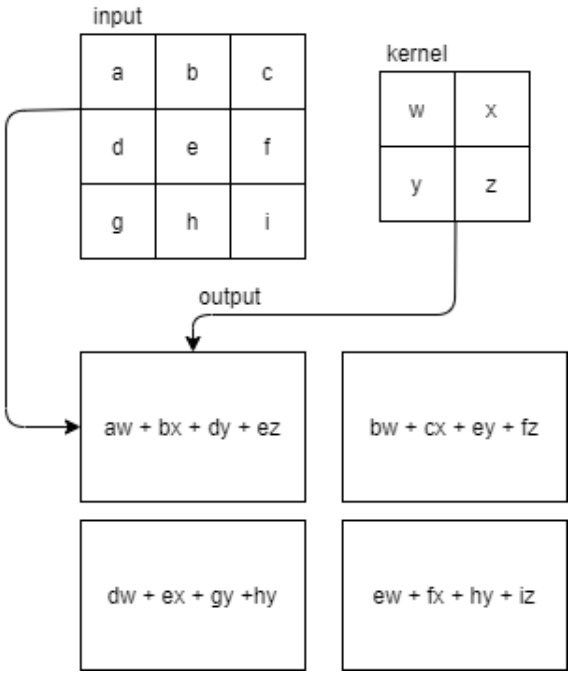


Figure 2.4: Simplified example of convolution where kernel slides on top of the input matrix. Starting position is up-left corner and ending position is the bottom-right corner.

2.3.2 Padding and stride

The shape of the input and kernel defines the shape of the output of the convolutional layer. Because the filter is often smaller than the actual input, after several convolution operations some information is lost at the edges of the original images.

With padding and stride, it is possible to determine the output size of the layer. [19, p. 237-239]

The padding adds extra pixels to the edges of the image and typically these added pixels are in the value of 0. This operation increases the image size and thus after convolution operation, the size of the input image is preserved. By using odd kernel size and padding with the same amount at each side of the image, spatial dimensionality can be preserved. [19, p. 237-239]

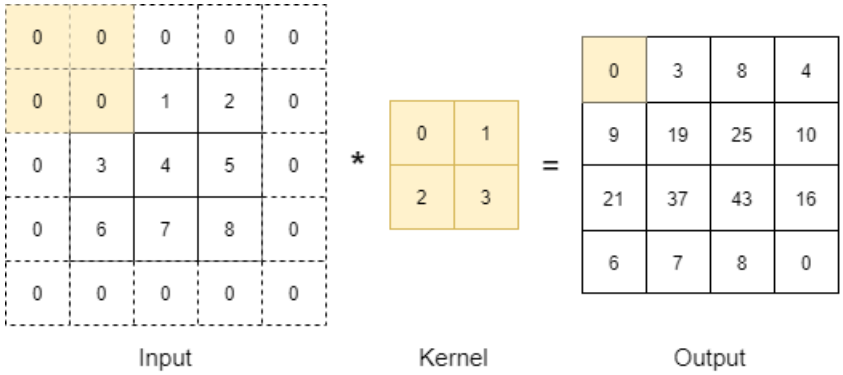


Figure 2.5: Convolution with padding.

The computational efficiency of the network or downsampling of the image can be affected with stride. Stride defines how many pixels are skipped in the image when kernel is moved to the next location. Stride size does not always have to be the same on the X-axis and Y-axis, and they can separately be defined. Using large stride size reduces resolution of the image [19, p. 240].

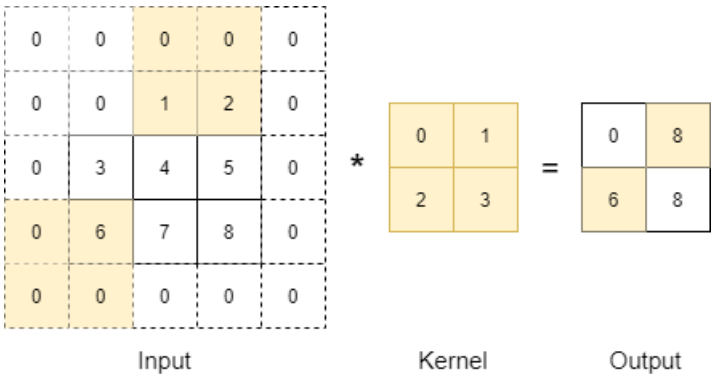


Figure 2.6: Convolution with stride of 3 and 2 for height and width.

2.3.3 Pooling

In convolutional neural networks pooling layers are used after convolutional layers. These pooling layers simplify the output from the convolutional layer by condensing feature maps. [14] The network obtains summary statistics of certain locations by using pooling. This process enables that small changes in input do not change the pooled values in outputs. In other words, representations become invariant to small translations of the input. This is a useful property if the goal is to identify if some specific feature is present in the picture but its location is irrelevant. [11, p. 342]

Pooling reduces memory requirements for storing the parameters and improves the computational efficiency of the network. Usually, in convolutional networks, fewer pooling units are used than detector units. This will lead to that the next layer of the network will have fewer inputs than its predecessor. The reduction of input size can result in improved statistical efficiency. [11, p. 342]

Simply put, pooling replaces the output of certain regions of the net with some function. Commonly used function is max pooling where function outputs the maximum value of a rectangular neighborhood. Also, the average of a rectangular neighborhood, the L^2 norm of a rectangular neighborhood, or a weighted average can be used as metrics [11, p. 339-342].

2.3.4 Sparse interaction

Convolutional networks have typically sparse interactions, opposite to fully connected networks where each input unit interacts with each output unit. Sparse interactions are accomplished with kernels that are smaller than actual input. This means that fewer parameters are stored and thus reducing memory requirements and improving statistical efficiency. Due to sparse interaction computing output requires fewer operations compared to a fully connected network. [11, p. 335]

2.3.5 Parameter sharing

The traditional feedforward network does not share parameters between computations. Meaning that each element in the weight matrix is considered only once when computing the output of a layer. However, convolutional networks use parameter sharing which refers to using the same parameter for more than one computation.

Instead of learning separate parameters for every location, the convolutional network learns a set of parameters. [11, p. 335-338]

2.3.6 Image preprocessing

Pal and Sudeep state that preprocessing image data is a vital step to archive valid classification accuracy[23, p. 1778]. Images can be preprocessed in many ways and some techniques can be combined to achieve different results.

In mean normalization brightness of the training set is normalized by calculating the mean along the data set and subtracted from each image [23, p. 1779]. Applying normalization helps to control gradients during the training phase when backpropagation is used.

$$X' = X - \mu \tag{2.17}$$

After the mean is normalized, the standardization can be applied. This can be performed by computing the standard deviation of the training samples and then dividing each image with the computed value. After this process, the mean and variance are normalized. [23, p. 1779]

$$X' = \frac{(X - \mu)}{\sum} \tag{2.18}$$

Where μ is mean vector computed over the given data and $\sum$ is the standard deviation across all the given data.

Convolutional neural networks can detect features based on the object's edges present in the image. Zero component analysis (ZCA) transformation makes edges more distinguishable in the image. [23, p. 1779] This transformation is also referred to as ZCA whitening. It is a linear transformation between random variables and known covariance matrix. ZCA whitening can be applied as follows:

Usually, in RGB image pixel values range from 0 to 255. These values can be scaled to obtain range [0, 1]

$$X' = \frac{X}{255} \tag{2.19}$$

After the scaling, the mean is normalized by (2.17). Singular values and the vectors of the covariance matrix are calculated and used to perform ZCA transformation. [23, p. 1779]

$$X_{ZCA} = U.diag(\frac{1}{\sqrt{diag(S) + e}}).U^T.X \tag{2.20}$$

Where diag(a) corresponds to a matrix with the vector a as a diagonal and 0 in all other cells, with U the left singular vectors, S values of the covariance of the normalized images, and the X the normalized data. e being the hyper-parameter called the whitening coefficient.

Pal and Sudeep tested mean normalization, standardization, and ZCA transformation with three convolutional neural networks, each having different architecture. The training set consisted of 40000 randomized 32x32 color images. As a baseline, all three networks were trained without preprocessing. With no preprocessing, 51-56$\%$ accuracy were achieved. Mean normalization resulted in 55-58$\%$ accuracy, standardization 63-66$\%$ accuracy, and ZCA increased to 64-68$\%$ accuracy. [23, p. 1780]

2.3.7 Image augmentation

Training a neural network for object detection requires large-scale dataset and usually obtaining this large-scale image dataset is a daunting process. Fortunately, with the image augmentation scale of the existing dataset can be expanded. Image augmentation makes a series of changes to training data while still retaining its important features. This produces more training data from a single image. For example, images can be cropped and rotated in different ways to achieve different positions of the object in the image. This can reduce the model's sensitivity to the object's position. Sensitivity to brightness and color can be reduced by adjusting brightness and color. [19, p. 537]

Chapter 3

Object detection

The ability to see and tasks related to vision are endogenous for humans and for some animals but not for computers. Humans can easily distinguish three-dimensional shapes of objects while machines require mathematical techniques to extract information about the shape and appearance of the object. [24, p. 3] Computer vision is an active research area that studies deep learning techniques and image processing to produce a diversity of applications [11, p. 452].

Object detection is a complex task compared to regular image classification because besides classification it demands object localization. Localization creates two primary challenges. First, there are numerous rough object locations to be processed. Second, these rough locations must be refined to gain more accurate locations. Usually, finding a solution to these problems requires sacrifice to speed, accuracy, or simplicity. [25, p. 452]

The general approach to this problem is to create a large set of candidate regions and then use CNN to extract features from them. After feature extraction, the regions are classified and the decision is then made whether the area contains a certain object. [22, p. 7]

3.1 Region proposal

Searching for an object at every image location and scale with sliding windows paradigm is computationally costly and inefficient [26, p. 391-392] and according

to Kong et al. object detection methods are moving from dense window sliding approaches to sparse proposal techniques [27]. High-quality detectors promote the development of object detection by reducing the number of windows that need to be classified [27, p. 845].

Current object proposal generators have a high recall, and they perform with efficiency as they use a modest number of candidate bounding boxes [26, p. 391-392]. In other words, the goal is to generate just enough regions that the true objects are found in the image.

Region proposal can be performed with a convolutional neural network and this technique is used in the experimental part of this thesis and is discussed in 3.2.3. Next, selective search and edge boxes are presented as they can be utilized in object detection architectures which are discussed in 3.2.

3.1.1 Selective search

The goal of the selective search is to generate class-independent object locations based on input data [28, p. 155]. As the scale of the objects and visibility of exact boundaries in images usually vary, the above-mentioned algorithm uses hierarchical grouping to overcome this problem. Hierarchical grouping uses a bottom-up approach where it starts from smaller locations and continues grouping until the whole image becomes a single region. The algorithm calculates similarities between neighborhooding regions and groups the two most similar regions. This process is repeated until the whole image becomes a single region. This way all scales of the objects can be captured. [28, p. 156]

Selective search has a diverse set of strategies for grouping regions together. Regions may form an object based on many features like color, texture, or size. Color of light or shading may affect how regions form and object. To take account for different conditions of lightning selective search performs hierarchical grouping in a variety of color spaces. In addition to color space, regions are grouped based on similarity measures based on color, texture, size, and fill. [28, p. 157-159]

3.1.2 Edge boxes

Zitnick et al. proposed that the object in images can be found based on their contour. The number of contours that are contained by the bounding box is indicative of the likelihood of the box containing an object. [26, p. 391]

Edge boxes use a Structured Edge detector to find an initial edge map. With this detector, the number of possible bounding boxes can be reduced. Edge groups are formed from neighboring edge pixels that contain similar orientations. The score for the bounding box is computed by summing the edge strength of all edge groups which are located inside the suggested box. [26, p. 393]

The sliding window approach is used to assess if the suggested bounding box contains an object. At each promising location, the score is calculated for the object being present. Locations with high scores are further refined using a coarse-to-fine search. [26, p. 393]

Zitnick et al. reported that with PASCAL VOC dataset they achieved over 96 percent object recall at an overlap threshold of 0.5 and 75 percent recall at an overlap of 0.7. [26, p. 393]

3.2 Convolutional object detection

Due to the progress of recent years in the field of convolutional object detection, it has made it possible to deploy object detection systems in consumer products. Some of these systems are fast enough to run on mobile devices. Running time, precision and memory usage are critical aspects of deploying real-time object detection in consumer platforms. For example, self-driving cars require real-time detection and mobile devices require low memory consumption. [29, p. 7310-7311]

In this thesis, the main focus is on region-based convolutional neural network (R-CNN) [30], Fast R-CNN [25] and Faster R-CNN [31], although it has been stated that Faster R-CNN tends to run slower than other cutting-edge convolution-based object detection systems but producing more accurate models. [29, p. 7315] However, if the grain and ear detection from drone images is not done in real-time (during flight), time criticality is not a requirement for the implementation of an object detection system. In this thesis, we focus on analyzing images afterward the flight, and not real-time detection.

3.2.1 Region-based convolutional neural network

Region-based convolutional neural networks (R-CNNs) published in 2013 by Girshick et al. [30] are an innovative approach to object detection. R-CNN divides an image into several regions which are then fed to CNN to extract features from each

separated region. After the feature extraction model predicts the category of each region and draws a bounding box around the found object [19, p. 581].

R-CNN consists of multiple stages of operations. On the input, the selective search is performed. This allows for selecting several proposition regions. Size, scale, and shape usually varies between these selected regions. After the region selection, pre-trained CNN is used in truncated form and it transforms proposed regions into the network's input dimension. CNN extracts the features from the input and these features are combined to the labeled category of each proposed region. These combinations are passed to multiple support vector machines (SVM) to perform training for object classification and trained SVMs determine which category example belongs. [19, p. 581]

A drawback of R-CNN is slow speed due to the model independently extracts the features of each proposed region. The problem multiplies as the number of proposed regions grows. This could cause thousands of computations for a single image. Usually proposed regions are overlapping in large measure and independent feature extraction results in repetitive computations. [19, p. 581-582]

3.2.2 Fast R-CNN

In 2015 Girshick published a method called Fast R-CNN [25] which performs the convolutional operation on the whole image instead of performing forward pass on each object proposal. This improves computational efficiency on images where object regions have a high degree of overlap [19, p. 582]. Object proposal regions are extracted using some external methods like selective search. Fast R-CNN takes an image and extracted proposal regions as input and uses several convolutional and max-pooling layers to produce a feature map. From the feature map, the region of interest (RoI) pooling layer produces a fixed-length feature vector. This is done for each object proposal. After the RoI pooling layer fully connected (FC) layers are used with fixed-length feature vectors as input which are produced by RoI pooling. At the final stage the FC layer branches into two separate output layers. The other output layer uses softmax function to estimate the probability for object classes and the other layers are used to output bounding box location using regression. [25, p. 1441]

3.2.3 Faster R-CNN

Region proposal network (RPN) is a convolutional neural network that uses an image as input and outputs set of object proposals. An object detection system called Faster R-CNN is composed of RPN and Fast R-CNN, where RPN and Fast R-CNN uses shared convolutional layers. In this composition, RPN tells the Fast R-CNN where to look for the objects. The popular term for this kind of network is neural network with attention. [31, p. 3-4]

Region proposals are generated by sliding small network over the shared convolutional feature map. Anchor boxes are used in sliding windows to capture multiple scales of the objects in the image. At each location, multiple region proposals are predicted and each of these sliding windows are mapped to a lower-dimensional feature which are fed to the box-regression layer and box-classification layer. [31, p. 3-4]

If maximum possible proposals for individual locations are denoted as k, then the regression layer has 4k outputs encoding the coordinates of k boxes. The classification layer outputs 2k scores that estimate the probability if an object appears in each proposal. These proposals are called anchor boxes. Each anchor is associated with scale and aspect ratio and anchors are fixed at the center of a sliding window. Ren et al. stated that they have used 3 scales and 3 aspect ratios, resulting in k = 9 anchors at each sliding position. For feature map of a size W x H (typically 2,400), resulting WHk anchors. [31, p. 3-4]

Faster R-CNN anchor boxes have two important properties. Firstly, anchor boxes and the functions that compute proposals relative to anchors, are translation invariant. Meaning, if the object is translated in the image, the Faster R-CNN should still be able to predict the location of the object. Secondly, Faster R-CNN can produce predictions in multiple object scales. [31, p. 4]

Training the RPN can be done utilizing backpropagation and stochastic gradient descent. Mini-batches are drawn from a single image that has an example of anchors. 256 samples of anchors are randomly chosen for mini-batch to compute the loss function. [31, p. 5]

Chapter 4

Experiment overview

In this thesis, an object detector pipeline for detecting cereal grains and ears is implemented. The implementation is done using a consumer computer and free open-sources resources. The results are evaluated using the metrics introduced in 4.3. The object detector pipeline in this experiment uses only one dataset containing oats, barley, and wheat images but the system is generic in the sense that with small changes it can be modified to identify other objects as well.

The dataset is divided so that the training dataset does not contain the same images as the testing dataset. This division is crucial because the intent is to evaluate how well the detector would perform on unseen images. This would reflect the situation where such an object detector is deployed in a real-world application, where the user could fly the drone over the cultivation while the detector would for example detect the foreign grains or count the number of wheat ears.

Building such a system that can detect cereal grains and ears could be used to monitor the quality of gluten-free cultures to detect foreign grains from oat fields. The problems related to pure-oat cultivation and foreign grains were described in 1.1.1. This experiment suggests an object detector that could be used in early detection of problem areas of cultivation.

For the development of the networks, Keras deep learning framework was chosen. With Keras API, the network layers are easy to implement, yet it offers enough potential to customization that the object detector can be implemented. Also, one selection criterion for the deep learning framework was that it has to support NVIDIA graphic card. Using TensorFlow and Keras, it is possible to utilize GPU on compu-

tations required for the training and testing of the network.

Some other possible frameworks and technologies were studied for the implementation of the object detector. Some of them proved to be not compatible with the available environment or too complicated to install, so they were ruled out.

4.1 Dataset

The dataset consists of drone overflight digital RGB images taken from an altitude of approximately 4 meters. Dataset was collected in summer 2019 from various locations. In this thesis images from three cereal plant fields are used to compile training and testing datasets. Both datasets consist of images of oats, barley, and wheat.

One pitfall of this dataset is that it does not contain images from the whole growing season. For a general object detector, it would require the dataset to contain images from different growing stages of the plants. Also, each field location was photographed on a single day which may affect the results.

4.1.1 Cropping

Original images are the size of 4864 x 3648 pixels. Due to the high resolution, one individual image contains numerous areas where visible grains occur. Given this fact, the images were cropped so that areas of 256 x 256 pixels were selected from multiple samples. The selection criterion for the choice of the region was considered to include as much variation as possible in the result. In the original images, there are multiple regions that vary in lightning condition, number of grains, occlusion of the grains, sharpness of the image, and the stance of the cereal crops. As a note, cropping an image does not change the size of the actual object in the cropped image.

4.1.2 Annotation

The main reason for selecting smaller regions from original images and not to use the whole image was the vast number of grains in one sample. Annotating images of cereal crops is a time-consuming and quite challenging process. In some cases, it is difficult to distinguish individual grains from each other or from the background

Those cropped regions where it is not possible to annotate objects unambiguously are discarded. Incorrect annotation of the objects can result in invalid results.

Oats Barley Wheat

Figure 4.1: Each object in the image is marked with a bounding box. The bounding-box consists of four coordinates that denote its location on the image.

Annotation was performed using LabelImg software, which is a free open-source graphical image annotation tool written with Python. The oats were annotated so that the individual grains were labeled with bounding-box and leaves and straws were not annotated. Marking the whole plant was not an option since from the aerial image it is not possible to distinguish the exact boundaries of the individual plants. Barley and wheat images are annotated so that the individual ears were marked. Also, with wheat and barley, the leaves and straws were left out of the process.

4.2 Object detector architecture

One limiting factor for the experiment was limited computing resources. The architecture of the network had to be chosen so that the training time was reasonable with a single computer with a single GPU. In addition to the training time, there was a need to consider how much graphics card memory could be allocated to the training and testing of the network. Adding more parameters to the network would consume more memory from the GPU. On the other hand, a model with too few parameters would not be able to learn more complicated features.

In this thesis, the network architecture and the adjustment of the parameters required ad hoc design and implementation. Meaning that decisions related to network structures were made based on the results obtained using test data during the development phase. Using testing data in the development phase creates a situation

where the results are partly biased. The reader must consider this fact when reading the chapter 6 where the results are presented.

Since there are many open-source examples of implementations of Faster R-CNN available, it was chosen to be the reference architecture of the object detector. Also, in previous studies, good performance of Faster R-CNN was reported [31, p. 7-11]. Keras implementation of the networks provided the opportunity for fast customization and prototyping.

There are pre-trained weights available for varying network architectures which could be used with Keras implementation of Faster R-CNN. Using these weights would defeat the purpose of this thesis since one of the goals was to determine if it is possible to train the network without using pre-learned weights and with a relatively small dataset.

4.3 Evaluation metrics

In this thesis, the performance evaluation of the object detection system is based on the same metrics as used in The PASCAL Visual Object Classes Challenge (VOC). [32]

The confidence score measures the probability of the anchor box containing an object. In Faster R-CNN, this score is predicted by the classifier. Intersection over Union (IoU) is the area of the intersection divided by the area of the union of a predicted bounding box and a ground-truth box. [32, p. 11]

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{4.1}$$

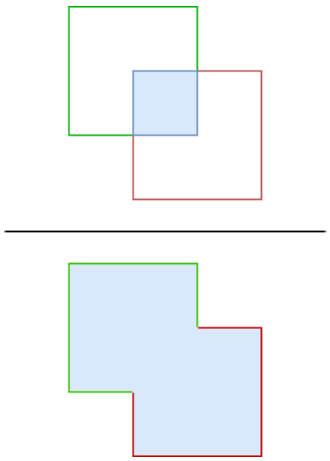


Figure 4.2: In the numerator, the blue area describes the intersection between the bounding box and the ground-truth box. In the denominator, the blue area describes the union between the bounding box and the ground-truth box.

Definition for true positive (TP), false positive (FP), and false negative (FN) are defined in [33]. True negatives (TN) can be ignored, and they are not included in metrics used in this thesis. For some objects, there are more than one predicted bounding box overlapping the ground-truth. For those predictions, we consider the prediction with the highest IoU as TP, and the rest predictions are considered as FP. This rule is derived from the PASCAL VOC 2012 metric. [33]

TP	A correct detection. Detection with IOU >= threshold
FP	A wrong detection. Detection with IOU < threshold
FN	A ground-truth not detected

Figure 4.3: The table explains how predictions are defined [33]

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

4.3.1 Precision-recall curve

When calculating the precision-recall rates of the object detector, we consider only one class at the time. This way the problem can be seen as a binary classification problem: is the object present in the predicted bounding box or not. These decisions can be represented in confusion matrix, where decisions are divided into true

positives, false positives, true negatives, and false negatives. Using this confusion matrix, we can plot the precision-recall curve, where recall is plotted on the x-axis, and precision is plotted on the y-axis. Precision measures a fraction of examples classified as positive that are truly positive and recall measures true positive rate which is the rate of correctly labeled positive examples. [34, p. 234]

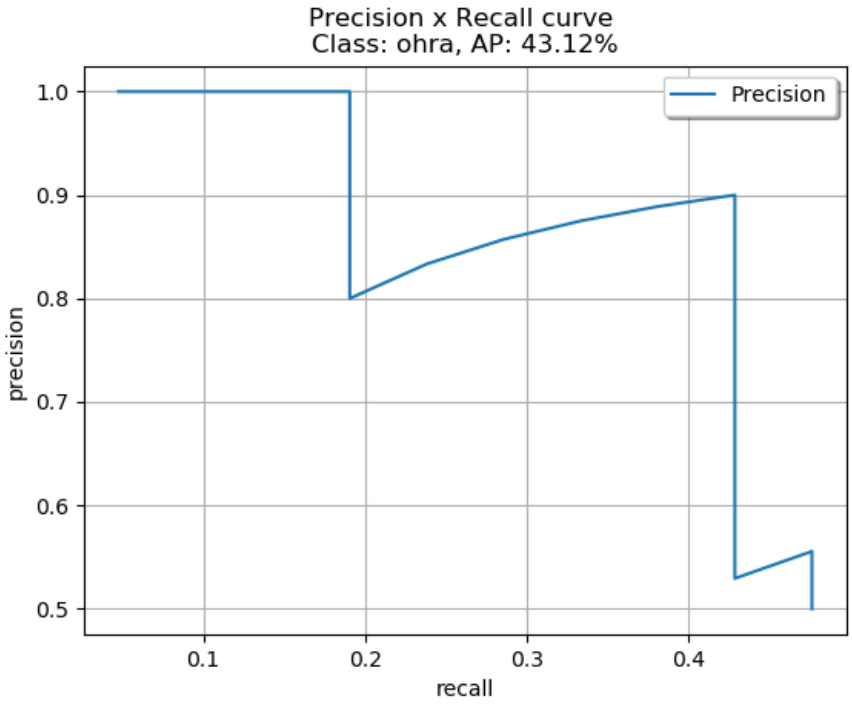


Figure 4.4: Example of precision-recall curve.

The precision-recall curve can be used to evaluate the performance of an object detector. The curve is plotted for each object class and detect can be considered good if its precision stays high when the recall increases. A poor object detector increases the number of detections to detect all ground-truth boxes, which increases the number of false positives and leads to lower precision. [33]

4.3.2 Average precision

Average precision (AP) is summarization of precision-recall curve 4.3.1. In VOC metrics, interpolated precision is used to calculate AP. $r_1, r_2, ...r_n$ is the recall levels at which the precision is first interpolated.

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1}) \tag{4.4}$$

In object detection problems there are usually more than one class. The calculation of AP involves only one class but calculating mean average precision (mAP), the average precision across all K classes can be obtained.

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{4.5}$$

Chapter 5

Experiment implementation

This chapter describes the implementation and training pipeline of the object detector. The implementation is based on the original Faster R-CNN, VGG-16 network, and various open-source GitHub repositories with Apache 2.0 license [35]. Apache license allows usage and modification of the original source code.

Since this experiment is not aiming to produce a real-time object detector, the time performance of the implementation is not critical. Utility components are not critically evaluated and can affect how fast the whole implemented system works. These components refer to those parts of the program which handles the tasks related to data processing.

The components that affect the network's detection capability have been implemented with the official TensorFlow machine learning platform and Keras neural networks abstract programming interface. The performance of the network is monitored with TensorBoard and the final evaluation is performed with metrics introduced in 4.3. A review of the results is presented in chapter 6.

5.1 Environment

The experiment was implemented with Windows 10 desktop computer with Intel i5-8600k 3,60Mhz CPU, GeForce GTX 1070 Ti 8,0 GB GPU, and 16 GBs of RAM. The object detector was implemented with TenforFlow 2.0.0 and Keras 2.3.1. The programming was done with Python 3.6, Anaconda virtual environment, and the

PyCharm 2019.3.1 IDE.

5.2 Network implementation

The original implementation of Faster R-CNN framework used VGG-16 [36] as base network [31]. In this experiment modified version of the VGG-16 is used to preserve computational resources and on the other hand to enhance the detection of the small objects. The layers are implemented with Keras 2D convolutional layers (Conv2D) and Keras max-pooling layers (MaxPooling2D)

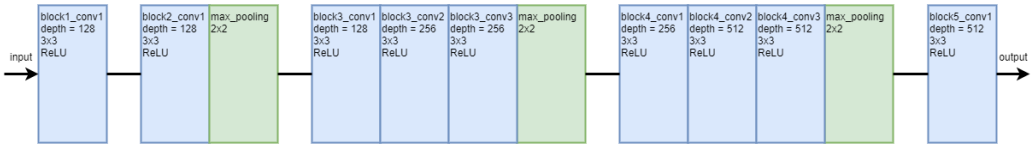


Figure 5.1: Illustration of base layers. Each individual blue box describes a convolutional layer. In this context, the depth is the channel number of the convolutional layer and the entry below it indicates the filter size. The green boxes reflect 2x2 max-pooling layers.

Since the VGG-16 network is used as a base network for the object detector and not stand-alone image classifier, last max pooling, fully connected and soft-max layers of the original VGG-16 are not implemented. Instead, we are connecting the last Conv2D block to the region proposal network and classifier to share the base layers with RPN and classifier. Also, the number of layers and depth is reduced. Only three max-pooling layers are used to preserve the feature map size which enables stride size of 8 px instead of 16 px which was used in the original implementation [31, p. 6]. This change helps identify oat grains since they often appear in dense groups in images, making the 16 px stride often too large.

5.2.1 Region proposal network

Region proposal network uses the last convolutional layer of the shared layers to generate region proposals by sliding a small network over the convolutional feature map [31, p. 3]. During implementation, it was noted that the performance of the RPN improved when two convolutional layers with 512 channels were used instead of one. In this experiment, the RPN is using two Conv2D layers with 512 output channels and a kernel size of (3,3).

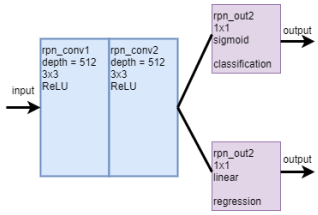


Figure 5.2: Illustration of RPN layers. Convolutional layers are illustrated as blue boxes and In this context, the depth is the channel number of the convolutional layer and the entry below it indicates the filter size. The purple boxes describe the outputs of the network.

RPN has two outputs the first being for the classification to predict if the object is present or not and second being the regression for the bounding box coordinates. Classification output is constructed from Conv2D layer and its input size depends on the number of anchor boxes. Since 3 different ratios and 4 scales are used, we get 3 x 4 = 12 anchor sizes. Regression output uses the input size of anchors * 4. For both outputs, we use (1,1) kernel sizes.

5.2.2 Classifier

The last step of the Faster R-CNN is to classify the proposed object regions and refine the bounding box coordinates. To do this we implement RoI pooling layer, fully connected layers, classification output, and regression output for the bounding box coordinates.

Since fixed-size input vectors are needed for the fully connected layers, RoI pooling is used to transform the regions of interest into fixed-size vectors. For this experiment, we output 7x7 regions. Larger regions caused memory allocation problems in the system and could not be implemented. In the original Fast R-CNN article, Girshick stated the implemented RoI pooling layer used max-pooling on the pooling region [25, p. 1441]. In this experiment, the max-pooling is substituted with the Tensorflow resize method, as it proved to be a more effective option with this pipeline.

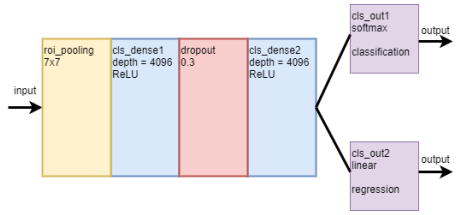


Figure 5.3: Illustration of classification layers where the yellow box denotes the RoI-pooling layer and the dense layers are illustrated as blue boxes. In this context, the depth is the unit number of the dense layer. The red box signifies the dropout with 0.3 rate. The purple boxes describe the outputs of the network.

Both fully connected layers are implemented with Keras Dense layers with 4069 units. Before the first dense layer, we flatten the input with Keras Flatten. The flattened layer gets the input from the RoI pooling layer.

Classification output is constructed with Keras Dense layer and has soft-max activation. This output predicts the class of the object. The input size of the layer depends on the number of classes used on the training. In this thesis, we use 3 classes for the cereal plants and 1 class for the background. For the bounding box coordinates, we use Keras Dense layer with an input size of 4 * (number of classes - 1) and with linear activation function.

5.2.3 Optimizer

For both RPN and classifier, the Adam optimizer from Keras optimizers library is used which is explained in 2.2.5. Other parameters are left to default values but the learning rate is set to 0.00001. Keras documentation states that if default values are used, their implementation follows those provided in the original paper [20].

5.2.4 Loss function

The loss function is implemented in four parts, which are at the final stage summed together. For the RPN classification loss, binary cross-entropy is used over all the selected anchors for minibatch. Binary cross-entropy is used since at this stage we are only interested if the box contains an object or not. Then, for the RPN bounding box coordinates smooth L1 loss is used. For the classifier classification loss, categorical cross-entropy is used and for the bounding boxes, the smooth L1 loss is used.

5.3 Pipeline overview

5.3.1 Image preprocessing and augmentation

First, the standard deviation is calculated over the training data. Second, the pixel intensity mean values are computed per color channel over the training data. Both the standard deviation and channel mean values are stored. At the training stage, when the image is randomly drawn from the training data set, preprocessing is applied. The first step in preprocessing is subtracting the color channel mean values from the image. The subtraction is made on a channel-by-channel basis. After the color channel subtractions, the pixel values are divided by the standard deviation and finally, values are scaled between 0 and 1.

Augmentation consists of random horizontal and vertical translation and random rotations. Both horizontal and vertical translations are added at 50% probability when the training sample is drawn from the dataset. For rotations, 0, 90, 180, and 270 angles are used. Transformations and rotations are performed so that the original ground truth values are adjusted accordingly.

5.3.2 Drawing the minibatch

Image is drawn from the training set and augmentation is added. After the augmentation, IoU is calculated for all possible anchor sizes and scales and anchors are labeled accordingly. Only 256x256 cropped images are used to train the network, even though it would be possible to use the varying sizes of images. IoU is calculated between ground truth boxes and anchor boxes. In this experiment, we set the threshold limit 0.7 for positive anchors and the other anchors are labeled as negative or neutral. If the IoU is larger than 0.3 but smaller than 0.7, the anchor is labeled as neutral and won't be used in training. Like in the original Faster R-CNN, 256 randomly selected sampled regions are chosen from all possible samples for the minibatch [31, p. 5].

5.3.3 Training the RPN

As mentioned above, RPN is implemented so that its input consists of a feature map of the base network's output. With the feature map, we input also the previously calculated anchor boxes. RPN outputs a set of region proposals with the probability that the object is present in the region. At this stage, objects are not classified.

The output is refined to regions of interest and all those regions which are not in boundaries of the feature map, are discarded.

5.3.4 Non-maximum suppression

The output of the RPN produces proposals that may overlap over the same object. To refine further the proposed regions, Non-maximum suppression (NMS) algorithm is used. In the original publication of Faster R-CNN is not stated how they implemented NMS, but Ren et al. reported that the NMS reduces the number of overlapping proposals for the same object while not harming the overall detection accuracy [31, p. 7].

In this experiment, NMS implementation is based on Tomasz Malisiewicz's vectorized version of the NMS. The overlap threshold for the NMS is set to 0.7 which is the same as used in the original Faster R-CNN NMS [31, p. 7]. We sort the proposed bounding boxes by the objectiveness score (the probability that the box contains object) and remove those which ratio of overlap exceeds the threshold set to 0.7. Ren et al. reported that after the NMS number of proposals was about 2000 [31, p. 7], but because of limited computational resources, we limit the number of boxes to 300.

5.3.5 Sampling from the RPN output

From the RPN output, the ratio between negative and positive regions are refined. With this dataset and this training pipeline, negative regions dominate the output of the RPN. This problem is mitigated by using only 4 regions instead of 300 [31] from the output, while trying to maintain a 1:1 ratio between positive and negative regions.

5.3.6 Classifier training

The classifier is trained with feature map outputted by the base layers and the region proposals of the RPN. As mentioned in 5.2.2, RoI pooling is used to get fixed-size feature vectors from the RPN output. At this stage, we classify objects in proposed regions and refine bounding box coordinates.

Chapter 6

Results

This chapter presents the results of the experiment. The loss value curves of the training the detector are presented, effects of the NMS are analyzed and precision-recall curves are presented and finally, a summary of the results is provided.

6.1 Training details

For training the network, 340 different images were used. 68 of the images were from oat fields, 153 from barley, and 119 from wheat. The images consisted of 4981 objects, of which 2400 were oats, 922 barley, and 1659 wheat. The variation in the number of objects can be explained by the fact that in these three plants the grains grow differently, especially if oats and the other two plants used in the experiment, are compared. Although fewer individual images of oats have been used than of the other two cereal plants, there are significantly more individual objects of oats. In this dataset, barley ears are on average considerably larger compared to a single oat grain or wheat ears.

Training data details		
Plant	Images	Objects
Oat	68	2400
Barley	153	922
Wheat	119	1659

Figure 6.1: The table shows how many images and objects the training data consists of.

Network training was performed in sessions so that the network was trained for 30 epochs and then the weight parameters were recorded. The length of a single epoch was defined as 1000 iterations, in each of which random samples of objects were selected. For this experiment, using this method, 180 epochs were run.

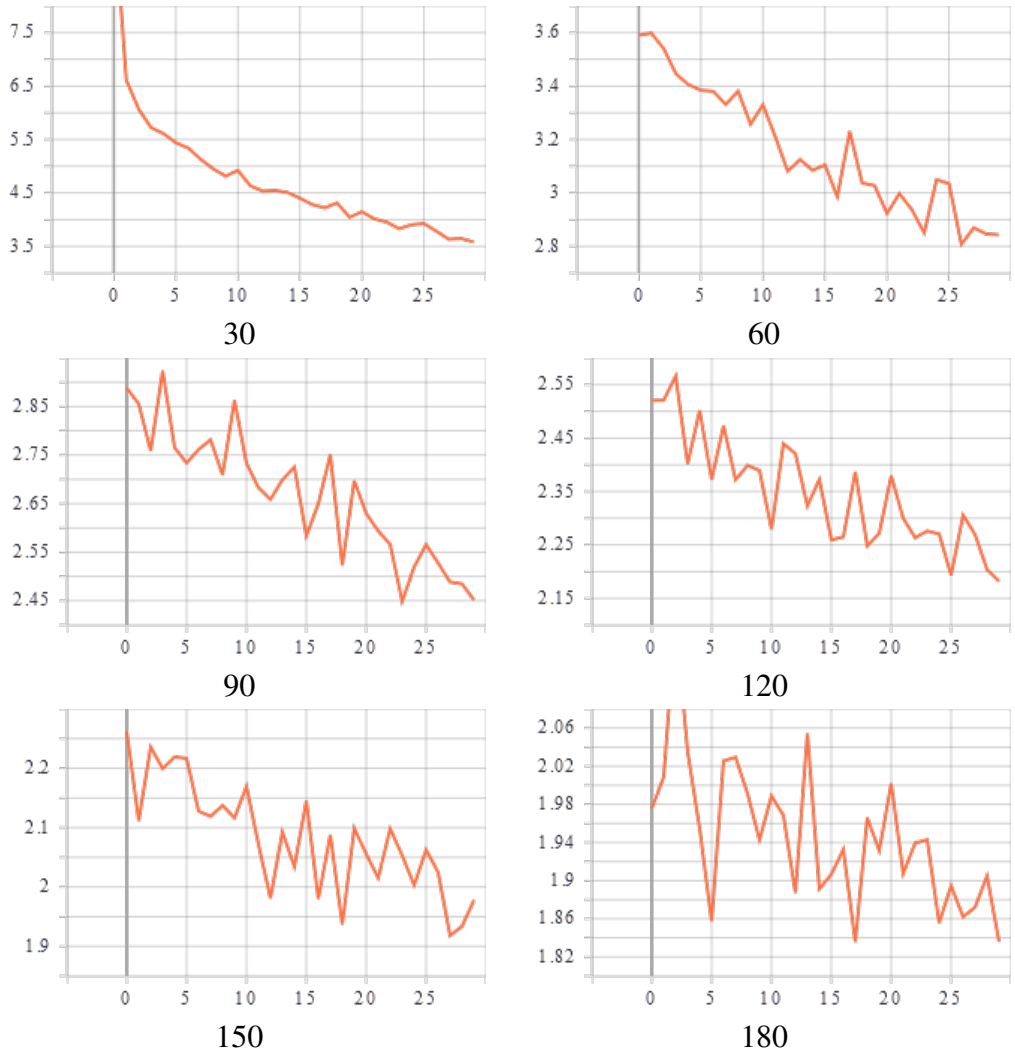


Figure 6.2: The changes in loss value for each training session are plotted. The x-axis indicates the number of epoch and y-axis the loss value. The number below each plot indicates the total number of epochs.

Figure 6.2 depicts the changes in loss value for each epoch batch during training time. The x-axis indicates the number of epoch and y-axis the loss value. As a note, the scale on the y-axis varies on different graphs, but the scale of the x-axis remains the same on each graph. Looking at the graph illustrating the first 30 epochs, it is notable that the loss value decreases rapidly after the first 10 epochs. After this point, the decrease in the loss value slows, and after 60 epochs curves start to show increasing fluctuations between epochs. Fluctuations are most likely explained by that the network is learning to predict one of the three classes better than the two others.

Despite the fluctuations, the overall trend of the loss decreases. From this, it can be

concluded that the network is starting to converge to some local minima. The effect between the last two training sessions is no longer significant if we are comparing the loss values. After 150 epochs loss has decreased to 1.90 and after 180 epochs loss has decreased to 1.83.

6.2 Evaluation details

Evaluating of the network was done using 92 images, of which 35 were images of oats, 31 of barley, and 26 of wheat. The total number of objects in the testing dataset was 1662 where 919 was oat objects, 347 barley objects, and 396 wheat objects. Attempts were made to compile the test data in a way that the annotation of the objects was as unambiguous as possible. Without proper annotation evaluating the performance is not possible. However, in some cases, clear borders of certain objects are not distinguishable and some errors in the annotation process might have occurred. This may have affected the evaluation process in small amounts.

Evaluation data details		
Plant	Images	Objects
Oat	35	919
Barley	31	347
Wheat	26	396

Figure 6.3: The table shows how many images and objects the testing data consists of.

For visualization purposes, the IoU threshold is set to 0.5, NMS to 0.2, and the probability threshold is set to 0.7. This results that only those bounding boxes are drawn of which IoU with ground truth box is over 0.5 and confidence of the prediction is 0.7 or higher. Also, the number of overlapping boxes is reduced by the NMS. Those boxes which are predicting the same object class and overlap more than 0.2 units in IoU are removed.

6.3 Effect of non-maximum suppression

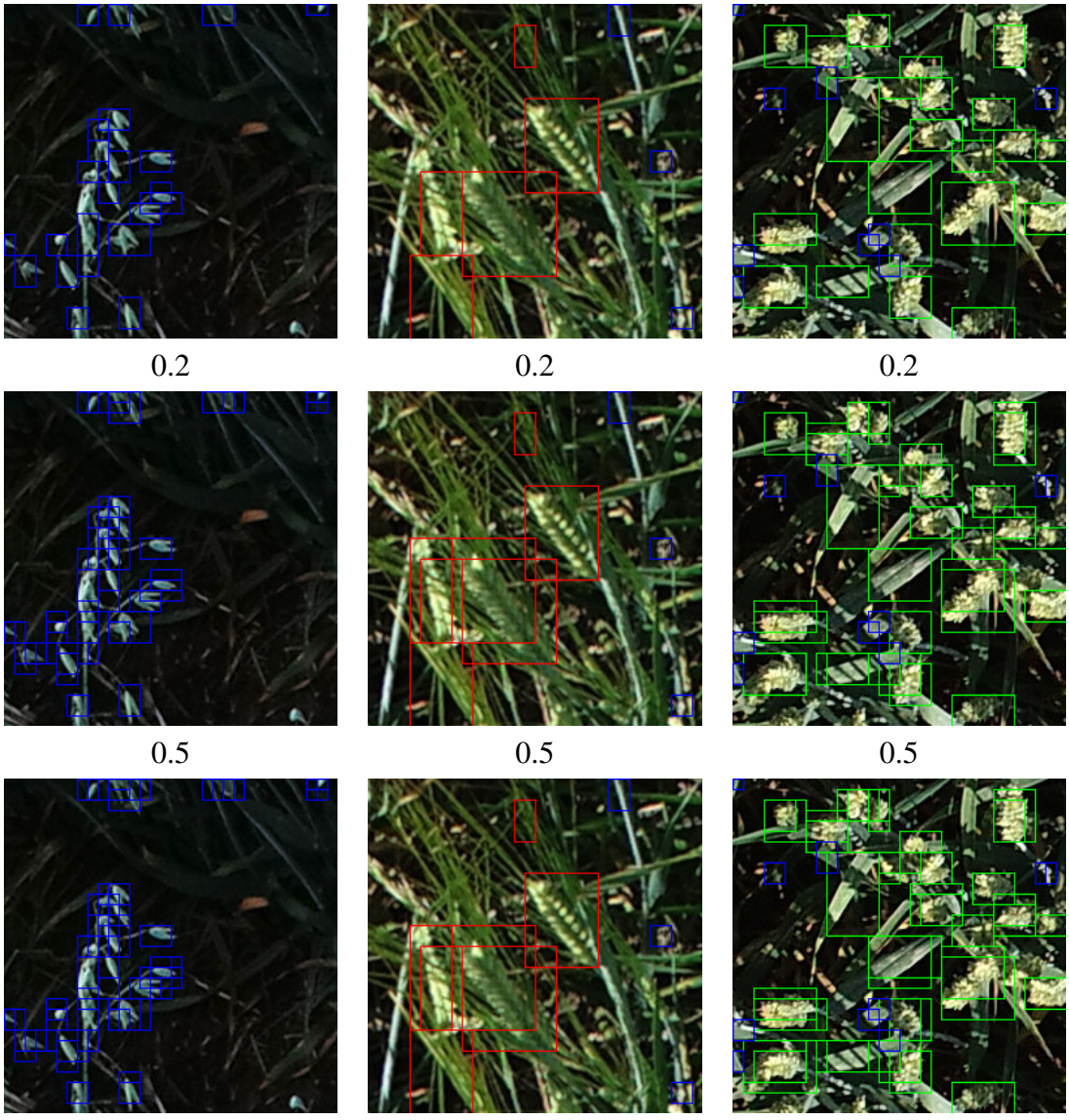


Figure 6.4: Illustration of three different NMS levels. In the first line 0.2 NMS was used, on the second 0.5 and on the third 0.8. The blue box indicates the prediction of oat, red box barley and the green box indicates wheat. For all predictions, 0.5 threshold for IoU and 0.7 threshold for probability is used.

The detector tends to produce multiple overlapping bounding boxes without NMS or when higher threshold values are used for NMS. With this dataset, lower threshold values provided better results and removed almost every overlapping bounding boxes but still retained true positives with reasonable rates. In figure 6.4 it is shown how increasing the threshold value increases the number of overlapping boxes and

on the other hand, decreasing the threshold value does not remove the true positives. Although in this experiment, using low threshold values for NMS produced good results, this might not be the case when using a different dataset. This sensitive NMS could remove true positives in those cases where objects are partially covered by each other but still clearly visible. Finding a perfect threshold value would require meticulous fine-tuning and experimenting.

6.4 Mean average precision

57.55% mAP was reached after training the detector for 180 epochs. Table 6.4 shows that there is a clear increasing trend on mAP after each training session. However, there is a notch in mAP after 150 epochs. After 150 epochs the average precision for oats has decreased significantly and this may be because the detector has started to prioritize the other two classes. However, after 180 epochs the situation has leveled off and the average precision for the oats has increased to 48.84. The detector reached 66.30 average precision for wheat after 150 epochs which were the highest average precision achieved during testing. Training the detector for 210 epochs lowered the mAP to 54,91% and after 240 epochs continued to decline to 54,68%. This might be an indication of the overfitting and so the last two training sessions were discarded.

Table 6.4 shows that the lowest average precision was obtained for the oats through all training sessions and this is most likely due to the nature of the data and the small size of the oat objects. The quality of the annotation and the images were the poorest for the images of oats. Also, the imbalance between the quantity of the test samples may affect evaluation results.

Table of AP and mAP				
Epochs	Oats	Barley	Wheat	mAP
30	35.47	48.08	51.58	45.04
60	47.63	56.34	52.02	52.00
90	42.57	58.44	60.90	53.97
120	43.97	55.66	65.76	55.13
150	37.69	57.14	66.30	53.71
180	48.84	60.57	63.24	57.55

Figure 6.5: Each row of the table is listed with the results of the training sessions. Epochs column indicates the total number of epochs. Oats, barley, and wheat column represent the average precision for each object class. mAP column is the mean average precision obtained after each training session.

6.5 Precision-Recall curves

Figures 6.6, 6.7 and 6.8 represents the precision-recal curves for the oats, barley and wheat. It is noticeable from the graphs, that detector achieves higher precision values at every recall threshold for barley and wheat than for oats. If performance is compared across all classes at recall threshold location 0.6, there is a clear distinction in precision levels. Detector achieved less than 0.5 precision at 0.6 recall threshold for the oats while for the barley it achieved slightly below 0.6 precision and for the wheat almost 0.7 precision.

Although the detector does not reach high precision levels at end of the curves, still it can be seen from the curves that the detector is able to detect objects from the testing samples and in fact, can distinguish classes from each other.

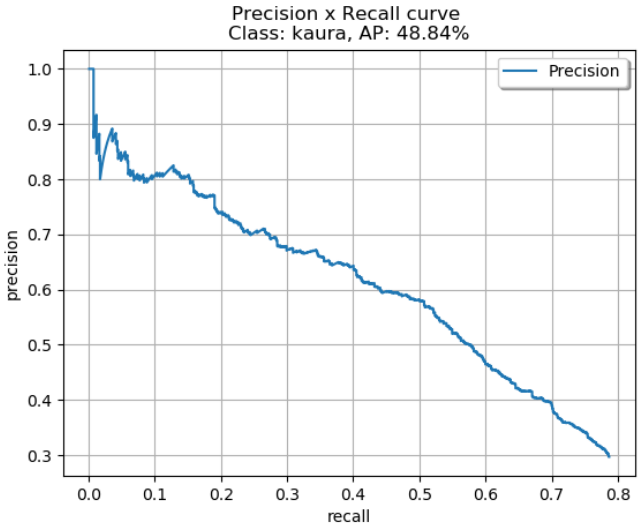


Figure 6.6: Precision-recal curve for the oat class.

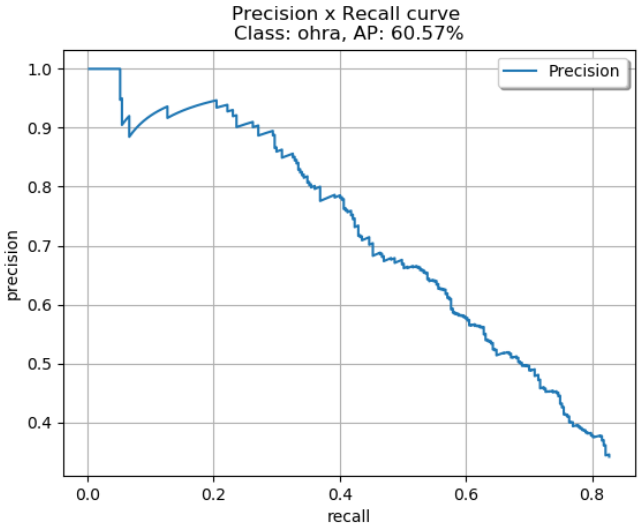


Figure 6.7: Precision-recal curve for the barley class.

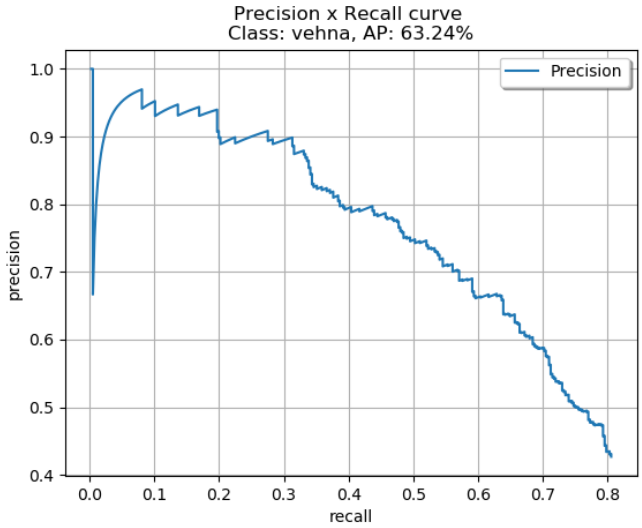


Figure 6.8: Precision-recal curve for the wheat class.

6.6 Summary of results

Although high average precision scores for all the classes were not reached, based on mAP and precision-recall curves, it can be said that the detector learned to detect different cereal plants and make distinction between them. The results support the idea that an object detector based on Faster R-CNN design can be used to solve problems related to precision-agriculture and detection of cereal grains and ears.

Also, this experiment shows that it is possible to implement and train an object detector using open-source resources, custom datasets, and limited computational resources. However, for fully comprehensive results it would require more training and testing data. The considerable challenge of this experiment was that the data was not annotated and it took great efforts to compile dataset even this size.

It is realistic to assume that with larger quantity and more varied data, such a system capable to supervise the quality of the gluten-free oat cultivation with aerial images and object detectors, could be implemented. This opens up the possibility that methods based on object detection can be combined more and more effectively with traditional farming and that the development of such methods with modern tools does not require enormous resources.

Chapter 7

Discussion

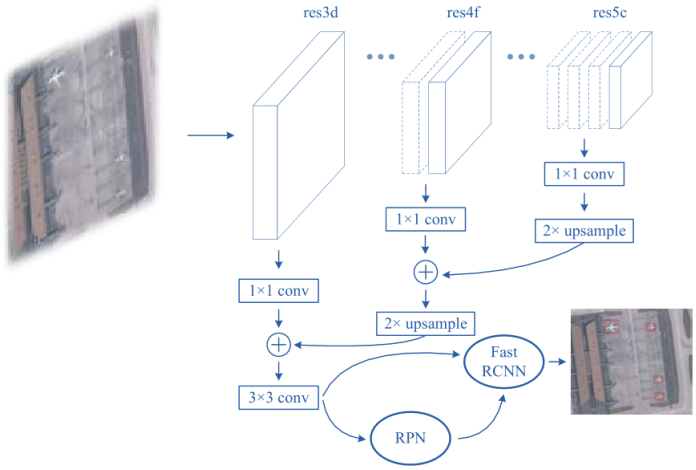
In this chapter, the results are further analyzed. Also, the problems of this experiment are being discussed and how those could be solved in the future. Discussion is based on referenced literature and how those techniques could be used to improve the implementation used in this thesis.

7.1 Small objects

Faster R-CNN object detector relies on convolutional neural networks to extract abstract feature representations of the objects and their locations. Using multiple convolutional layers and pooling layers results in down-sampled feature maps. With large objects, this does not cause significant problems as in these cases objects occupy a major portion of the image. [37, p. 2] However, with small objects down-sampling feature maps causes problems.

During the experiment of this thesis, it was noted that using 4 pooling layers at the base layers of the detector caused problems in detecting oat grains. In some cases, ground-truth box for the single oat grain can be less than 16 x 16 pixels, and most often oat grains reside side by side or even in some cases overlap in images. This of course is a major problem if the resulting stride for the detector is 16 pixels when using 4 pooling layers [31, p. 6]. By removing one pooling layer enables that the feature map is not scaled down so that the final stride is 16 pixels but 8 pixels instead. However, this has immediate effects on memory consumption and training times since the RPN and classifier has to work with larger feature maps.

Ren et al. designed an object detection model which is based on Faster R-CNN and modified ResNet-50 [38] as the backbone [37, p. 3]. The main idea was to use lateral connections between convolutional layer blocks and merge the up-sampled feature outputs by element-wise addition. 3 x 3 convolutional layer produces the final feature map and degrades the aliasing effect of up-sampling. [37, p. 3]



[37, p. 4]

Figure 7.1: Illustration of the modified Faster R-CNN.

The model was tested on the dataset which consisted of images of ship and plane objects. The size of the object varied between 10 x 10 pixels and 100 x 100 pixels. Using their dataset and modified Faster R-CNN with 10^2, 40^2, and 100^2 anchor scales, they reached 78.9% mAP. Using the original version of the Faster R-CNN with the original anchor scales 128^2, 256^2, and 512^2 they achieved 66.6% mAP. [37, p. 8]

The results clearly indicate that using this method improves the detection of small objects. Adapting a similar kind of lateral connection between layer blocks and up-sampling the feature maps could improve the detection of small oat grains.

7.2 False positives

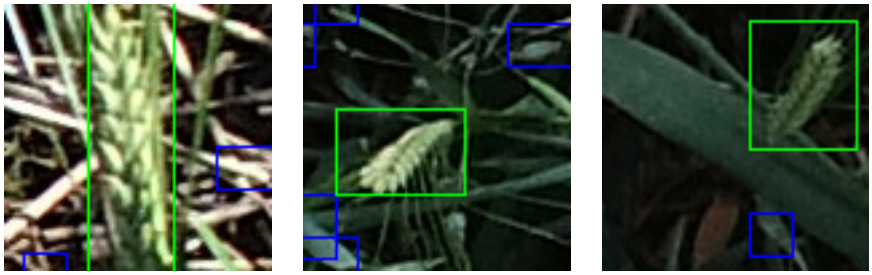


Figure 7.2: Barley ears falsely identified as wheat ears. The image also shows false positives for the oats (blue boxes).

In some cases, the object detector incorrectly classified barley ears as wheat ears. The training data is unbalanced in a way that it contains more wheat objects than barley objects which can be seen from the table 6.1. This may lead to false positives since barley and wheat have some similar features.

Ren et al. stated that they had a similar kind of imbalance in their dataset when conducting their experiment. Dataset consisted of 5216 ship images and 706 plane images. To address this, they applied a balanced sampling strategy, where the aim is to draw training samples from the dataset as uniform as possible within an epoch regarding classes. [37, p. 5-6] This strategy could be implemented to mitigate the problems associated with data imbalances in the dataset used in this thesis. The intuition is, if the training data consisted of the same number of examples of barley and wheat, the amount of miss classification of barley ears as wheat ears should decline.

Another way of looking at this problem of incorrect classifications is that since in some images the ears of these plants resemble each other, the downgrading effect of pooling layers might cause false positives. In the experiment of this thesis, three max-pooling layers are used in the base layers of the detector which means that the feature maps are downgraded three times before they are passed to RPN and classifier. In this process, some crucial features might be lost which could differentiate the objects from each other.

Otsuzuki et al. proposed regularized pooling which could be used to substitute max-pooling. They have stated that the max-pooling may over-compensate for essential differences between similar classes. [39, p. 1-2] In regularized pooling, the pooling operation is regularized to fit the characteristics of actual deformations of the object. This is done by taking the average of max value directions in the neighboring

kernels in a window which leads to that a non-maximum value can be selected and over-compensation does not occur. [39, p. 3]

Regularized pooling was experimented with convolutional network based on VGG [36], MNIST [40] and EMNIST [41] datasets [39, p. 7]. Otsuzuki et al. reported that the learning convergence is faster when using regularized pooling instead of max-pooling and also the excessive deformation compensations were suppressed [39, p. 8].

The strategy of using balanced sampling and regularized pooling might have the desired effect of overcoming the problem of false positives on miss classifying the barley ears as wheat ears. Ren et al. made a note in their article that their sampling strategy is cost-effective and easy to implement [37, p. 6]. Given this fact, this strategy could be implemented and used in the environment run by a consumer computer.

7.3 Network depth

During the implementation and training of the network, it was experimentally found that using the original VGG-16 architecture as a backbone for Faster R-CNN did not give good results. After decreasing the number of layers and also the channels per layer, the network started to converge faster and performance increased. However, finding the optimal number of layers is difficult using only one computer and training one instance of a network at the time. Experimenting on this matter would require a system that allows training multiple instances simultaneously without depleting computing time or resources from each other.

One explanation for this phenomenon is that the network simply does not converge in a reasonable time and the number of epochs used in experimenting was not sufficient or the dataset does not contain enough information. It is a known fact that regarding deep neural networks that too many parameters in hidden layers may cause overfitting when the training data does not contain all the necessary information to train all the neurons in the hidden layers. On the other hand, increasing the number of hidden layers may increase the training time beyond the limit that the training would be possible in practice.

He et al. stated that deep networks may be exposed to degradation problem. Increasing the number of network layers causes accuracy to saturate and degrade rapidly. Their experiments show that adding more layers to suitably deep networks leads to

a higher training error and it is not caused by overfitting. A deep residual learning network is suggested to encounter the degradation problem. The main idea is to allow the network to skip connections or skip over the layers which solve the problem where deep models have difficulties in approximating identity mappings by multiple nonlinear layers. [38, p. 770-773]

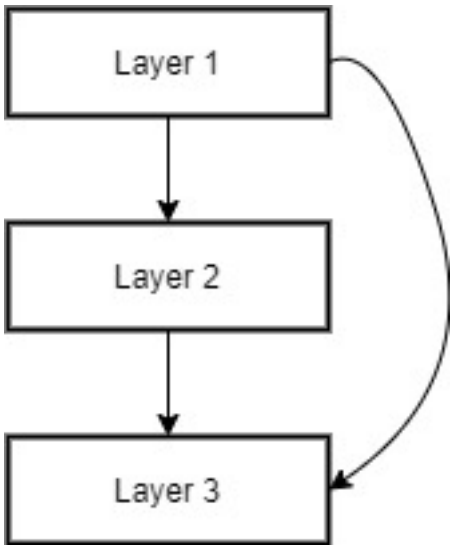


Figure 7.3: Illustration of a residual layers.

7.4 Synthetic data

As noted earlier in this thesis, the amount of data was one of the most limiting factors in the training of the object detector. Data augmentation offers limited opportunities for artificial expansion of the dataset, but there are other ways that have been studied and proven to work for expanding dataset. The training dataset can be expanded with computer-generated synthetic images.

Rozantsev et al. presented a system that could produce 3D models of objects of interest and use them as training data for object detector [42, p. 25]. Using a sliding window approach, convolutional neural network, and their Unmanned Aerial Vehicle (UAV) dataset, Rozantsev et al. reported that their model reached 0.85 average precision without synthetic images and with synthetic images, the model achieved 0.89 average precision. [42, p. 31]

Data annotation proved to be a very time-consuming process in this experiment because the images used in training the network contained a particularly large number of objects. Because of this, only a limited number of images could be prepared for

the training of the network. This resulted that the objects in images appeared with similar kind of backgrounds and lighting conditions. Tremblay et al. developed a pipeline that produced synthetic images with objects of interest appearing in different lighting conditions and with random backgrounds [43, p. 1082]. They used a domain randomization technique where generated objects are placed in random locations and random camera angles are used. Also, lightning conditions are randomized, and random geometric shapes were added to these scenarios. All objects were automatically annotated by implemented pipeline and Tremblay et al. reported that the pipeline is capable of outputting 1200×400 images with annotations at 30 Hz. Using synthetic training data created by the pipeline and Faster R-CNN object detector, they achieved 78.1 average precision.

Georgakis et al. studied a different approach for creating synthetic data. Instead of using 3D modeling, they used images of real environments and cropped objects. The main idea was to create a pipeline that can produce synthetic training data from real images of different environments and blend objects of interest in the images. They reported that when using 50% real data and 50% synthetic data, Faster R-CNN performance increased 1.3%. [44]

These studies show the potential of using synthetic data and automated pipelines to create data when training object detector. The referenced studies evaluated their methods on Faster R-CNN which supports the conclusion that the synthetic data could be used to train the object detector presented in this thesis. Generating synthetic images of whole cereal plants, grains, and ears would expand the training data set significantly and would help to train a more general model.

Chapter 8

Conclusions

On basis of the referred studies on Faster R-CNN and the experiment conducted in this thesis, it has been shown that this type of object detection system is suitable to solve problems related to precision-agriculture and detecting cereal grains and ears. The original Faster R-CNN design struggles to detect small objects which are common in data related to agriculture and aerial images, thus in that sense, the Faster R-CNN needs re-design. This thesis suggests that using fewer max-pooling layers than original VGG-16, improves the detection of small objects. However, this also increases need for more computing power since the object detector has to work with larger feature-maps. Studies [37] and [8] show that ResNet architecture provides better results than VGG-16 with Faster R-CNN. In light of this fact, substituting VGG-16 with modified ResNet suggested in [37] could improve the results.

Compiling a full-scale dataset for object-detection problems is a difficult and time-consuming process. The type of experiment conducted in this thesis would greatly benefit from using readily available large datasets. However, it was noted that there was no freely available dataset that would be suitable for this experiment. Lack of the right form of data seems to be a recurring difficulty related to object detection problems and this is also reported in [9] and [42]. 7.4 presented a few ways to solve these problems and studying more these methods is important since the demand for more and better quality data is ever-growing.

From the point of view of pure oat cultivation, the results obtained from the studies look bright. Object-detection designs based on convolutional methods have reached

high average precision on agricultural data that they would have a substantial impact on detecting foreign grains from the oat fields. A study conducted by Liu et at. based on Faster R-CNN and ResNet detected maize tassels with 91.51% average precision from 15 meters altitude promises good future for automatic field inspections with unmanned aerial vehicles [8]. With near real-time detection performance of Faster R-CNN design combined with the accuracy of methods suggested in [8], [37], [39], and [38] would greatly benefit the quality control of the pure-oat cultivation. With these suggested methods it is credible that the financial losses due to contamination of crops and manual activity related to the inspections could be reduced.

References

[1] Konstantinos G Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, 2018.

[2] Diego Inácio Patrício and Rafael Rieder. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. *Computers and electronics in agriculture*, 153:69–81, 2018.

[3] Michael Gooding. Chapter 18 - the effects of growth environment and agronomy on grain quality. In Colin Wrigley, Ian Batey, and Diane Miskelly, editors, *Cereal Grains (Second Edition)*, Woodhead Publishing Series in Food Science, Technology and Nutrition, pages 493 – 512. Woodhead Publishing, second edition edition, 2017.

[4] Esko K Janatuinen, Pekka H Pikkarainen, Tarja A Kemppainen, Veli-Matti Kosma, Ritva MK Järvinen, Matti IJ Uusitupa, and Risto JK Julkunen. A comparison of diets with and without oats in adults with celiac disease. *New England Journal of Medicine*, 333(16):1033–1037, 1995.

[5] Tricia Thompson. Oats and the gluten-free diet. *Journal of the American Dietetic Association*, 103(3):376–379, 2003.

[6] Satafood Kehittämisyhdistys ry. Puhdaskauran tuotanto-ohje 2018 ,gluteenittomasta viljelykierrosta erikoistumisvaihtoehto tiloille -hanke.

[7] Kinnusen Mylly. Puhdaskauran viljelyohjeet kinnusen myllyn sopimusviljelijöille, 2019. Version 5.

[8] Yunling Liu, Chaojun Cen, Yingpu Che, Rui Ke, Yan Ma, and Yuntao Ma. Detection of maize tassels from uav rgb imagery with faster r-cnn. *Remote Sensing*, 12(2):338, 2020.

[9] Mads Dyrmann, Søren Skovsen, Morten Stigaard Laursen, and Rasmus Nyholm Jørgensen. Using a fully convolutional neural network for detecting locations of weeds in images from cereal fields. In *International Conference on Precision Agriculture*. International Society of Precision Agriculture, 2018.

[10] Kadir Sabanci, Ahmet Kayabasi, and Abdurrahim Toktas. Computer vision-based method for classification of wheat grains using artificial neural network. *Journal of the Science of Food and Agriculture*, 97(8):2588–2593, 2017.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[13] Nikhil Buduma and Nicholas Locascio. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media, Inc., 1st edition, 2017.

[14] Michael A. Nielsen. Neural networks and deep learning, 2018.

[15] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, Inc., USA, 4th edition, 2008.

[16] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[17] 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.

[18] Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. Multilayer perceptron: Architecture optimization and training. *IJIMAI*, 4(1):26–30, 2016.

[19] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. https://d2l.ai.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks

from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[22] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[23] K. K. Pal and K. S. Sudeep. Preprocessing for image classification by convolutional neural networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 1778–1781, May 2016.

[24] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[25] Ross Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[26] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.

[27] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[29] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[31] 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.

[32] 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.

[33] Rafael Padilla; Sergio Lima Netto; Eduardo A. B. da Silva. Survey on performance metrics for object-detection algorithms. *International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020. https://github.com/rafaelpadilla/Object-Detection-Metrics.

[34] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[35] Apache Software Foundation. Apache license 2.0, 2004. https://www.apache.org/licenses/LICENSE-2.0.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Yun Ren, Changren Zhu, and Shunping Xiao. Small object detection in optical remote sensing images via modified faster r-cnn. *Applied Sciences*, 8(5):813, 2018.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[39] Takato Otsuzuki, Hideaki Hayashi, Yuchen Zheng, and Seiichi Uchida. Regularized pooling. *arXiv preprint arXiv:2005.03709*, 2020.

[40] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[41] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[42] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, 2015.

[43] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977, 2018.

[44] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. *arXiv preprint arXiv:1702.07836*, 2017.