Weed detection based on Convolutional Nerual Network

Jian Wu(xcb479), IT and Cognition

April 9, 2018

Abstract

This is the final report for one free topic project registered in Department of Computer Science, University of Copenhagen. The report mainly researches on the application of deep learning convolutional neural network on simple objects recognition task, and comparing the results with classical image processing methods. All code can be checked on GitHub repository.

1 Introduction

Pesticide regulations and a relatively new EU directive on integrated pest management create strong incentives to limit herbicide applications[6]. In Denmark, several pesticide action plans have been launched since the late 1980s with the aim to reduce herbicide use[10]. One way to reduce the herbicide use is to apply site-specific weed management, which is an option when weeds are located in patches, rather than spread uniformly over the field. So segmentation of weed area from the whole crop area is important. From Søren's work[11], it is possible to resolve this problem based on classical color analysis and texture analysis. However, with deep learning applied widely on many objects recognition tasks and image segmentation task, we want to research whether it is possible to do weed detection with deep learning tool, Convolutional Neural Network instead of the classical image processing. CNN for recognition of weed area is to get enough training area which can be taken by RGB cameras from one same attitude. Then cutting these images to many small patches. We label the patches manually. With labeled patches, we can train one CNN model using big scale data and get optimal parameters. Finally, the model can do thistle detection in any image.

The objective of this report is to demonstrate CNN is possible to do weed detection in general images and how CNN works for thistle detection task. This paper includes a short review of previous work, a description of training data, methodology applied to pre-process data and learning model design, results of model, example visualization of conv-filters and output from conv-layers.

2 Previous Work

2.1 Thistle detection based on classical image processing

The amount of previous work on weed detection is based on classical color image processing. In Søren's paper[11], they attempted to address this by color analysis and texture analysis.

They focus on detecting green thistles from yellow mature cereals. In practice, the two colors may be close and changing over time to place. For some specific situation, shape analysis should be added. They did color normalization for all images and then detected the color of each $1m^2$ area based on yellow cereal distribution, which was suggested by central limit theorem.

2.2 ConvNet layers

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing, so it becomes more and more popular in images recognition. The most classical CNN model in TensorFlow[1] is for RGB images across 10 categories(airplane, automobile, bird, cat, etc.) CNN is now mainly applied on image segmentation[13] and image classification[12].

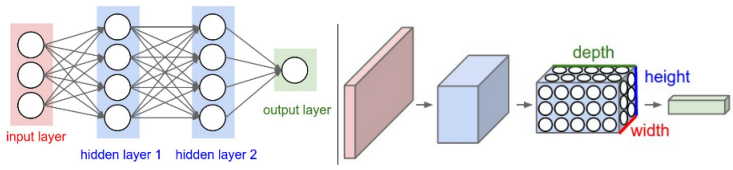


Figure 1: Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.

3 Data description

The training images are taken for two kinds of crops including barley and wheat, from different altitudes(20m, 30m, 50m). We focus on the barley images from 30m altitude.

All image for training and validation are taken from 30m altitude. The size of each RGB image is 3000×4000. Then we cut the original color image to

30×40 image patches and each size of patch is 100×100. For all image patches, they have been labeled with weed(1) or crop(0). Then we use these labeled cut image as training input. Some examples are given in Figure 2, 3.

Figure 2: The original image taken from 30m attitude.

Figure 3: The left one is $100 * 100$ weed patch, while the right one is $100 * 100$ crop patch.

4 Methodology

4.1 Pre-processing

Since our pictures are taken from different weather and different angle. So the light may have significant influence in the images for training and test. Converting it from RGB space to HSV space is a good idea for the origin images to be normalized, which is suggested by [11]. After transforming the representation to HSV. Here, the effect is concentrated both in saturation component and the value component, whereas the hue component is hardly affected. So we should focus on normalizing saturation component and the value component. We choose to correct both saturation S and value V using a linear fit to each

image, i.e.:

$$S(x,y) = \frac{\overline{S}}{a_s + b_s x + c_s y} S(x,y)$$

The same,

$$V(x,y) = \frac{\overline{V}}{a_v + b_v x + c_v y} V(x,y)$$

where $\overline{S}$ and $\overline{V}$ are global averages of saturation and value, and where the denominators show planar fits to the local averages of saturation and value. An example of a correction is shown in 4

(a) The original image will uneven illumination

(b) Compensated illumination

Figure 4: Comparing pre-processed image and original image.

4.2 CNN model

Due to the great performance of AlexNet[9] on image classification, I choose AlexNet as my learning model here. The most classical AlexNet model has nearly 60 million parameters, which is out of memory for normal students' development environment. So I simplify AlexNet structure here. My CNN has around 30 million parameters. The model is consists of thfee convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final softmax layer. To reduce overfitting in the full-connected layer I applied dropout method that proved to be very effective. In order to consider some unexpected situations, I follow data augmentation rules(random rotation, normalization for all data, etc.) mentioned in [9].

Classical AlexNet[9] works well in multiple classification task, like plants species recognition[4], marine animals recognition[3] etc. However, the labels for all image patches are only weed(1) or crop(0). I simplify the classical AlexNet to three convolutional and two functional layers. The first convolutional layer is 64 $3 * 3$ filters with one max pooling layer. Both the second and third convolutional layer are 128 $3 * 3$ filters with one max pooling layer after the third conv-layer.

The CNN structure is displayed in Figure 6 and Figure 5 describe specific parameter information about experiment ConvNetwork model.

4.2.1 Function layers

For the function layers, since there are just two classes of label, crop or weed, I use logic regression function here. Set that $y = 1$ when the label is crop. For $X_i = (x_{i1}, x_{i2}, ...x_{id})$

$$h(X_i) = \sum_{j=1}^{D} w_j x_{ij} + w_0 = W^T X_i + w_0$$

So,

$$\hat{y_i} = P(y = 1|X_i) \frac{1}{1 + e^{-h(X_i)}}$$

4.2.2 Training algorithm and optimization

During the back propagation phase, the model parameter is trained by the stochastic gradient descent (SGD) algorithm[2], with the categorical cross-entropy loss function as optimization object. The SGD can be expressed as follows:

$$\delta_x = w_{x+1}(\sigma^{'}(w_{x+1} \cdot c_x + b_{x+1}) \circ \text{up}(\delta_{x+1}))$$

$$\Delta w_x = -\eta \cdot \sum_{i,j} (\delta_x \circ \text{down}(S_{x-1}))$$

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 100, 100, 64)	1792
activation_1 (Activation)	(None, 100, 100, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 50, 50, 64)	0
conv2d_2 (Conv2D)	(None, 50, 50, 128)	73856
activation_2 (Activation)	(None, 50, 50, 128)	0
conv2d_3 (Conv2D)	(None, 50, 50, 128)	147584
activation_3 (Activation)	(None, 50, 50, 128)	0
max_pooling2d_2 (MaxPooling2	(None, 25, 25, 128)	0
dropout_1 (Dropout)	(None, 25, 25, 128)	0
flatten_1 (Flatten)	(None, 80000)	0
dense_1 (Dense)	(None, 10)	800010
activation_4 (Activation)	(None, 10)	0
dropout_2 (Dropout)	(None, 10)	0
dense_2 (Dense)	(None, 2)	22
activation_5 (Activation)	(None, 2)	0

Total params: 1,023,264
Trainable params: 1,023,264
Non-trainable params: 0

Figure 5: Specific information about parameters of CNN.

where δ_x is sensitivity, w_{x+1} is multiplicative bias, $\circ$ indicates that each element is multiplied, up is upsampling, down is downsampling, Δw_x represents the weight update of the layer, and η is the learning rate. And I define the cross-entropy loss function as:

$$L_i = -\log(\frac{e^{f_i}}{\sum_j e^{f_j}})$$

where f_j is the j-th element in the classification score vector f. After some preliminary training experiments, the base learning rate is set to 0.001, which

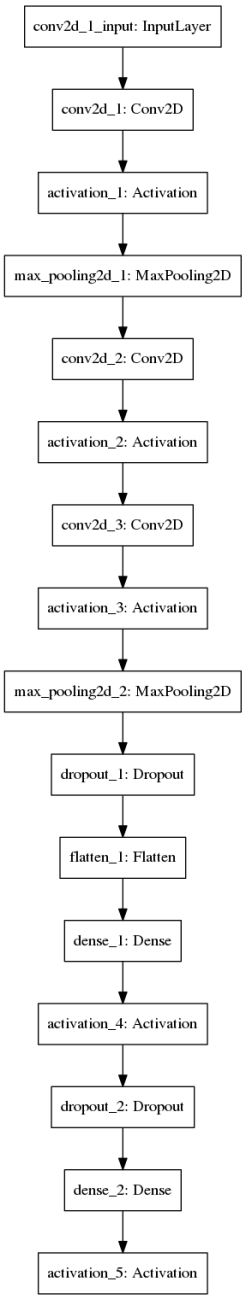


Figure 6: The entire structure of CNN, which consists of three convolutional layers(with 64, 128, 128 filters separately) and two functional layers.

is gradually reduced at each epoch. The decay rate is 10^{-6} and the momentum is 0.9.

5 Experiment

5.1 Data generation

There are 38 images with labeled patches, totally 45600 labeled patches. Due to the limit of running time and experiment hardware, I randomly choose 5000 patches for the training and validation, which includes 3000 for training and 600 for validation.

5.2 Experiment Settings

For convolutional neurual network, I follow the pipeline of AlexNet[9]. Due to the limitation of the local development environment, I reduced the number of filters in every layers and the amount of conv-layers. In the full connected layer, I restricted the node number to 10, which should be enough for only binary classification.
The model implementation is based on the open source deep learning framework Keras[5] and image processing package OpenCV[8]. All the experiments were conducted on a Ubuntu 16.04 Linux server with a 3.40 GHz i7-3770 CPU (16 GB memory).
During the training, random transformations and normalization operations were configured on all training image data during training, which has been proved to be helpful to improve the final accuracy[9].

6 Experiment Result

6.1 Accuracy for train and validation

After nearly 40-hours training, the final two models was generated. One model was trained directly based on original images, while the other one was created using color-normalized image data. Tensorboard[1] was used to record the training process. From Figure 7, which displays the accuracy on train set and validation, it is described that the accuracy for training set is nearly 100% in both with and without pre-processing. However, performance of two models on validation set is similar, 13% lower than that on training set.

Table 1: The final score for validation set and test set

	Validation	Test
With Color preprocessing	99.8%	86.6%
Without Color preprocessing	99.6%	86.2%

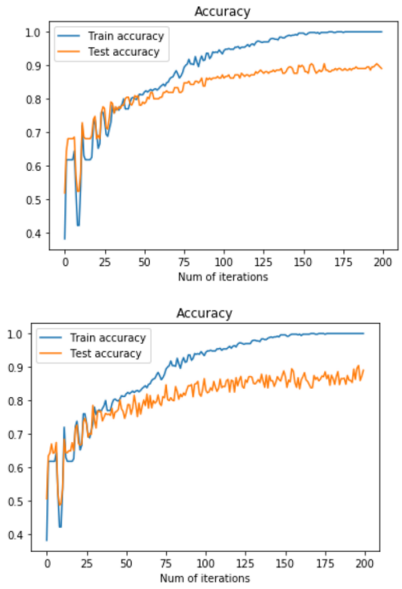


Figure 7: Left: Accuracy without pre-processing, Right: accuracy with pre-processing

6.2 Visualization

Filter and output visualization is helpful to make deep understanding on the convolutional neural network. Some random example filters from all 3 conv-layers are displayed in Figure 8.

In order to see how input images travel through all CNN structure. We choose one original image shown in 9 for inputing experiment. Then we plot some example output images respectively from all 3 convolutional layers .

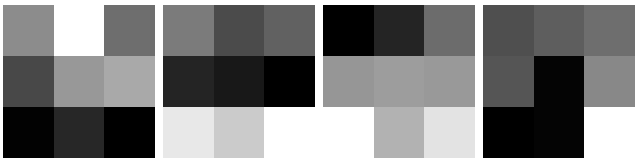
6.3 Results

After seeing how CNN model works and outputs on each layer, I input entire original image and let model work on detecting it each patch. The result is described in Figure 11.

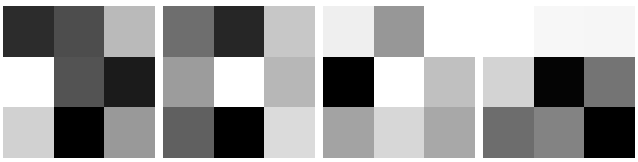
7 Conclusion

7.1 About preprocessing

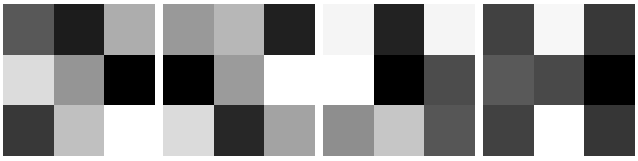
Firstly, as results shown in Figure 7 and Table 1 based on the results gotten from the images with preprocessing and images without preprocessing , there



(a) Four random convolutional filters from the 1st conv-layer



(b) Four random convolutional filters from the 2nd conv-layer

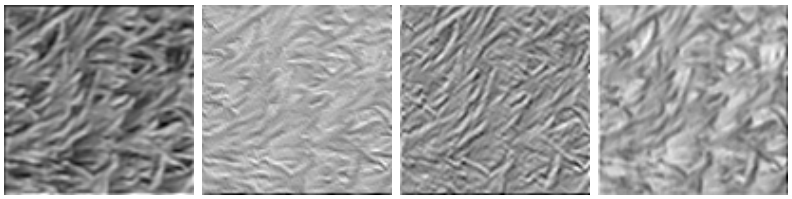


(c) Four random convolutional filters from the 3rd conv-layer

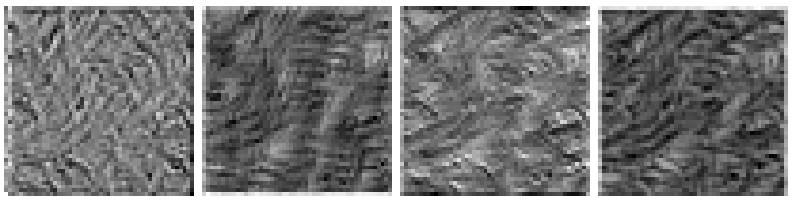
Figure 8: Some example convolutional filters from the all convolutional layers

Figure 9: The origin image for input to CNN

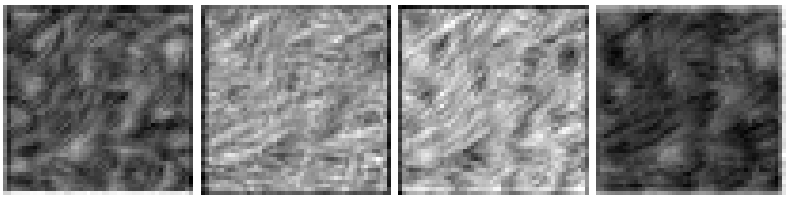
is nearly no differences between these two method. It can indicate that CNN require minimal preprocessing. So, in many CNN application, the preprocessing will be superfluous. For this case, we just need to train CNN model with original image.



(a) Four random output examples from the 1st Convolutional layer



(b) Four random output examples from the 2nd Convolutional layer



(c) Four random output examples from the 3rd Convolutional layer

Figure 10: Original input image displayed in Figure 9. Some random example outputs are shown above. The original input size is 100×100, the output from the 1st conv-layer will be still 100×100. However, due to the max-pooling layer following the 1st conv-layer, the output from the 2nd and 3rd conv-layer will be 50×50.

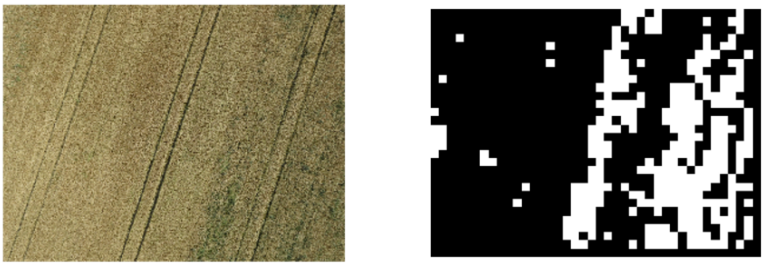


Figure 11: The left one is original image, the right one is final detection based on trained model, with thistles areas marked as white.

7.2 About the CNN structure

Since there are just two classes and all the batches are not complicated, the CNN structure should not be complex. At first, I set 4 convolutional layers and 64, 128, 128, 256 convolution filters. However, overfitting was caused by the complex structure. The accuracy for training set is significant high, nearly 100%. But for validation set, the accuracy is just 76%. Which is much lower than accuracy for training set. So, we redesign the CNN with more simpler structure. As a result, we should consider whether the images are complicated and the number of classification is big. For simple images across few categories, CNN should not be designed too complex. It is important to keep the learning model suitable, which can make the learning process more efficient and enough for specific tasks.

8 Future work

Although this experiment can indicate that deep learning convolutional neural network can work well in weed/crop classification, the accuracy for the validation is not as good as result from classical image processing method[11]. This part mainly discuss how to improve the CNN model based on experience and feedback from this experiment.

Firstly, we plan to use more training data. Currently, because of limitation of hardware both in memory and running speed, I only used 5000 image patches for training. It seems still not enough for building a general and perfect model for weed detecting in any field.

Secondly, another method for improving the model may be to upgrade the hardware. It is popular to train the CNN model on GPU[14]. Since the computing ability of GPU is more powerful and it can be possible to fast the learning speed and find more optimizing parameters.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Zheng Cao, Jose C Principe, Bing Ouyang, Fraser Dalgleish, and Anni Vuorenkoski. Marine animal classification using combined cnn and hand-designed image features. In *OCEANS'15 MTS/IEEE Washington*, pages 1–6. IEEE, 2015.

[4] Qiang Chen, Mani Abedini, Rahil Garnavi, and Xi Liang. Ibm research australia at lifeclef2014: Plant identification task. In *CLEF (Working Notes)*, pages 693–704, 2014.

[5] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[6] EU Directive. 128/ec of the european parliament and of the council of 21 october 2009 establishing a framework for community action to achieve the sustainable use of pesticides. *EU, Brussels*, 2009.

[7] Hervé Goëau, Alexis Joly, Pierre Bonnet, Souheil Selmi, Jean-François Molino, Daniel Barthélémy, and Nozha Boujemaa. LifeCLEF Plant Identification Task 2014. In L. Cappellato, N. Ferro, M. Halvey, and W. Kraaij, editors, *CLEF2014 Working Notes. Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014*, CEUR Workshop Proceedings, pages 598–615. CEUR-WS, 2014.

[8] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015.

[9] 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.

[10] Bo Melander, Nicolas Munier-Jolain, Raphaël Charles, Judith Wirth, Jürgen Schwarz, Rommie van der Weide, Ludovic Bonin, Peter K Jensen, and Per Kudsk. European perspectives on the adoption of nonchemical weed management in reduced-tillage systems for arable crops. *Weed Technology*, 27(1):231–240, 2013.

[11] Søren I. Olsen, Jon Nielsen, and Jesper Rasmussen. Thistle detection. In Puneet Sharma and Filippo Maria Bianchi, editors, *Image Analysis*, pages 413–425, Cham, 2017. Springer International Publishing.

[12] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[13] Rahimeh Rouhi, Mehdi Jafari, Shohreh Kasaei, and Peiman Keshavarzian. Benign and malignant breast tumors classification based on region growing and cnn segmentation. *Expert Systems with Applications*, 42(3):990–1002, 2015.

[14] Balazs Gergely Soos, Adam Rak, Jozsef Veres, and Gyorgy Cserey. Gpu powered cnn simulator (simcnn) with graphical flow based programmability. In *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, pages 163–168. IEEE, 2008.