Weed detection based on CNN

Jian Wu(xcb479)

October 2017

1 Introduction

Pesticide regulations and a relatively new EU directive on integrated pest management create strong incentives to limit herbicide applications. In Denmark, several pesticide action plans have been launched since the late 1980s with the aim to reduce herbicide use. 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 recognition of weed area from the whole crop area is important and classical image processing method with color analysis make it possible. In Søren's paper[1], they resolve this probelm with color nalysis and texture analysis, which can recognize the weed area. However, with the hot of deep learning, we try to resolve this problem with deep learning tool, Convolutional Neural Network, instead of the classical image processing. The approach for CNN 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 patches. We label the patches manually. With labeled patches, we train one CNN model and get optimal parameters. Finally, we can use these model to recognize any image.

2 Previous Work

The amount of previous work on weed detection is based on classical color image processing. In Søren's paper[1], they attempted to address these kinds of problems by color analysis including texture analysis. However, CNN makes many classifying images issues become more simple. The most classical CNN model in tensorflow is for RGB images across 10 categories(airplane, automobile, bird, cat, etc.)

3 Data description

There are many images taken from different attitude for different kinds of crops(including barley and wheat). We focus on the barley images from 30m attitude. The all image for training and testing is taken from 30m attitude.

The size of each color image is $3000 * 4000$. The weed or crop patch is cut from original image, whose size is $100 * 100$. There are 1900 sample patches with label.

Figure 1: The original image taken from 30m attitude.

Figure 2: The left one is $100 * 100$ weed patch, whhile the right one is $100 * 100$ crop patch

4 Methodology

4.1 Preprocessing

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[1]. 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,

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

4.2 Learning model

4.2.1 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. Since the labels for all images are only

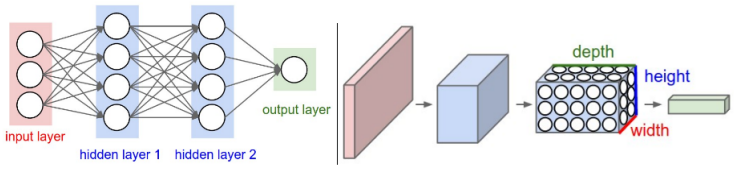

Figure 3: Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.

weed or crop. So I design the CNN architecture as simple as possible.
I design one simple two-layers CNN structure. The first convolutional layer is 16 $5*5*3$ filters with one max pooling layer. The second convolutional layer is 8 $5*5*3$ filters with one max pooling layer. Since there are $100*100$ pixels for each picture, max pooling layer is necessary for shortening the training time.

4.2.2 Function layers

For the function layers, since there are just two classes of label, crop pr 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,

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

4.2.3 Cost function

I use,

$$f(x) = \frac{1}{C}\sum_{i=0}^{D} w_i^2 + \frac{1}{N}\sum_{i=1}^{N}(1-y_i)\log(1-\hat{y}_i) + y_i\log\hat{y}_i$$

here. In order to avoid the overfitting problem, I import $\frac{1}{C}\sum_{i=0}^{D} w_i^2$ here. I set $C = 100$

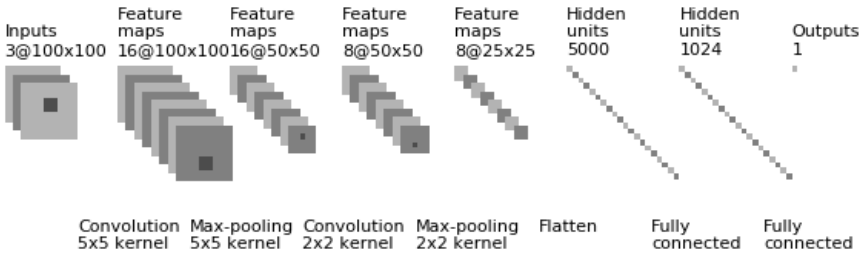

Figure 4: The entire structure of CNN, which consists of two convolutional layer and two func layers

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 Accuracy for train and validation

From the Figure showing the accuracy on train set and validation set. We can obtain that the accuracy for training set is nearly 100% in both with and without Preprocessing. The accuracy on valiation for two methods is always the similar trend and similar final value.

Methods	Accuracy on training set	Accuracy on validation set
With Color preprocessing	99.8%	86.6%
Without Color preprocessing	99.6%	86.2%

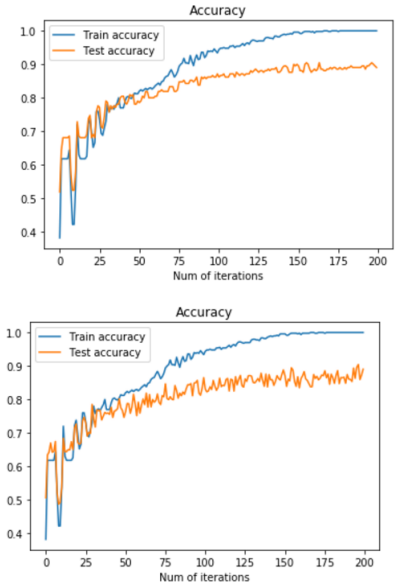

Figure 5: Left: Accuracy without preprocessing, Right: accuracy with preprocessing

5.3 Visualization

For each ConvNet layer, we visualize randomly 3 or 4 filters with origin image input. Shown in Figure 6 and Figure 7

5.4 Results

Weeddetect is currently implemented by Python and Tensorflow. Based on 5000 patches for the training and CNN structure mentioned aboove. It takes nearly 10 hours for training on i7 CPU. After saving the model parameters, it just

Figure 6: The origin image for input to CNN

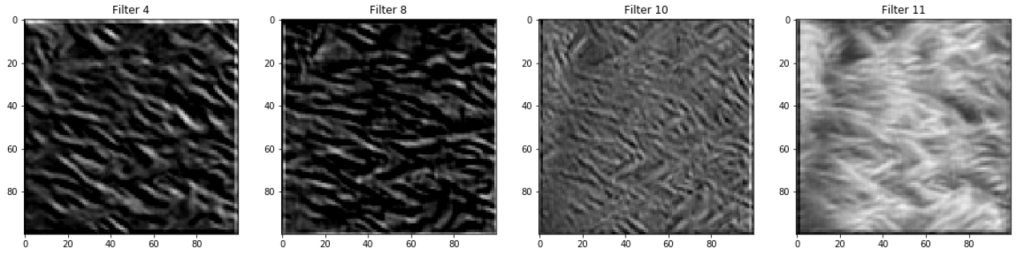

Figure 7: Randomly chosen 4 filters in the first convolutional layer

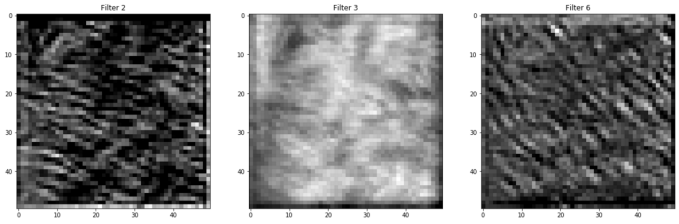

Figure 8: Randomly chosen 3 filters in the first convolutional layer

takes 1s to recognize the origin image.All of the code for this experiment can be viewed on my GitHub

6 Conclusion

6.1 About preprocessing

Firsly, based on the results gotten from the images with preprocessing and images without preprocessing, there is nearly no differences between these two

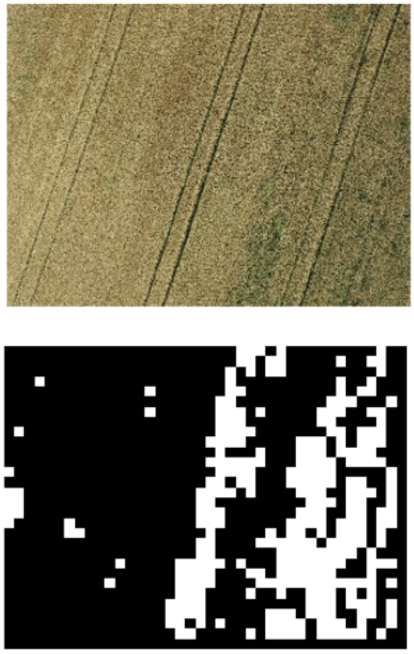

Figure 9: The upper one is original image, the below one is Final $1m^2$ -resolution detection with thistles areas marked as white.

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.

6.2 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 3 convolutional layers and 32 convolution filters for each ConvLayer.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, when we do the CNN structure design, we should consider whether the images are complicated and the number of labels is big. For simple images across few categories, CNN should not be designed too complex.

7 Future work

For this experiment, the accuracy for the validation is not as good as that got from classical image processing method mentioned in [reference]. So there are still a lot of things which needs to be improved in the further work. Firstly, we plan to use more samples for training with one GPU or a better CPU, which make it possible to train all the training set.
Secondly, we plan to improve the CNN structure, which should include some processing within the CNN, like randomly rotation, batch normalization, which can make CNN work better.

References

[1] Søren I.Olsen, Jon Nielsen, and Jesper Rasmusen *Thistle Detection*. In Scandinavian Conference on Image Analysis (pp. 413-425). Springer, Cham.