Convolutional Neural Network Application to Plant Detection, Based on Synthetic Imagery

Larry Pearlstein
Mun Kim
Warren Seto
Department of Electrical and Computer Engineering
The College of New Jersey
Ewing, NJ USA
e-mail: pearlstl@tcnj.edu

***Abstract*—Deep convolutional neural networks (DCNN's) have shown great value in approaching highly challenging problems in image classification. Based on the successes of DCNNs in scene classification and object detection and localization it is natural to consider whether they would be effective for much simpler computer vision tasks.**

Our work involves the application of a DCNN to the relatively simple task of detecting weeds in lawn grass. We looked at the effects of the choice of CNN hyper-parameters on accuracy and training convergence behavior. In order to obtain a large labeled set of interesting data we generated realistic synthetic imagery. Since our problem is somewhat constrained we were able to run thousands of training experiments and do accurate estimation of the probability density function of the convergence rate.

Our results suggest that the use of realistic synthetic imagery is an effective approach for training DCNNs, and that very small DCNNs can be effective for simple image recognition tasks.

Keywords—convolutional, DCNN, deep learning, computer vision

I. Introduction

The problem of controlling undesired plant species while promoting the development of desired crop plants dates back to the dawn of agriculture. Modern weed control generally relies on the indiscriminate application of chemical herbicides. This approach is expensive – it is estimated that billions of dollars are spent annually in the US on herbicides for commercial farming. Beyond the financial costs there are serious concerns about the impact of chemical herbicides on human and animal health as well, as on the health of desired crop species. Furthermore it appears that the repeated use of herbicides in an area actually gives rise to the emergence of herbicide-resistant weeds, which tends to negate the beneficial effects of the chemicals [1].

The challenge is well known to many homeowners, who struggle with control of weeds in lawns (see Figure 1) while balancing concerns over health risks to children and pets, and contamination of groundwater. There have been proposals to address this problem based on robotic weeding devices using computer vision, but an effective commercial solution has yet to be demonstrated.

A number of studies focusing on detecting weeds in lawns have been carried out based on the use of hand-crafted features, such as difference moments [2], Gabor wavelets [3], moments of enhanced images [4], combined morphological filters and moments [5] and local binary patterns [6]. The features were classified based on heuristic functions [2][3][4][5], Three-Layer Perceptron [3], Bayes analysis [4] or Support Vector Machine [6]. A study was conducted using a Deep Convolutional Neural Network, but was limited to isolated laboratory obtained leaf images [7].

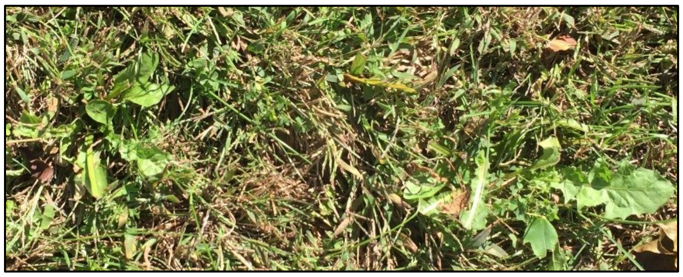

Figure 1: Example of Weeds in Home Lawn

Deep convolutional neural networks (DCNN's) have shown great value in approaching highly challenging problems in image classification. Based on the successes of DCNNs in scene classification and object detection and localization it is natural to consider whether they would be effective for much simpler computer vision tasks.

Our work involves the application of a DCNN to the relatively simple task of detecting weeds in lawn grass. We looked at the effects of the choice of CNN hyper-parameters on accuracy and training convergence behavior. In order to obtain a large labeled set of interesting data we generated realistic synthetic imagery. Since our problem is somewhat constrained we were able to run thousands of training experiments and do accurate estimation of the probability density function of the convergence rate.

We based our DCNN on AlexNet, which achieved notoriety for its outstanding performance in the 2012 Imagenet Large Scale Visual Recognition Challenge[8]. We trained our network with synthetic images, which were generated in an attempt to realistically model natural images. We started with a relatively large network (many convolutional features) and progressively pared down the network size until its ability to model the training set was impaired.

II. Methodology

1. Classification Framework

We set out to select among two hypotheses:

Hypothesis H0: image contains no weeds

Hypothesis H1: image contains weeds

We used a DCNN with two outputs, with one output representing the estimated probability of H0 and the other H1. The output with the higher level resulted in deciding for the associated hypothesis.

2. DCNN

We used the Berkeley Caffe tools [9] to train and test a DCNN modeled after AlexNet [8]. Similar to AlexNet our DCNN had five convolutional layers and three fully connected layers. Our input images were 256 x 256 pixels, and we used similar max-pooling and stride-based downsampling in the convolutional layers. We employed the rectified linear non-linearity, dropout training and cross-entropy loss function.

The key differences between our network and AlexNet were that we used grayscale input images, we decided among only two classes and we used far fewer features in our convolutional and fully connected layers. A comparison in DCNN hyperparameters between our network and AlexNet is given in Table 1.

Table 1: DCNN Hyperparameters – Comparison Against AlexNet

Layer	1 CNV	2 CNV	3 CNV	4 CNV	5 CNV	6 FC	7 FC
AlexNet	96	256	384	384	256	4096	4096
Present Study	4-64, all identical					4-128	16-128

3. Synthetic Data

We generated a set of 9,000 synthetic images. Software was developed to produce realistically complex occlusions and variations in leaf number, size, color and orientation in images simulating grass (representative of H0) and grass plus weeds (representative of H1), as shown in Figure 2. Two LMDB databases were created for use with Caffe; one with 8,000 images for training and the other with 1,000 images for testing. In each of these two databases, half of the images were representative of H0 and the other half were representative of H1. The order of the images in the database used for training were scrambled. Mean files were computed and stored.

4. DCNN Training

The entire training database of 8000 synthetic images was used in each training epoch. Training consisted of application of 100 epochs, for a total of 800,000 image-based gradient updates. Testing was performed at the end of each epoch.

Prior to training the DCNN weights were initialized randomly, according to the Caffe 'xavier' rule, which was derived based on [10]. Many training runs were attempted for

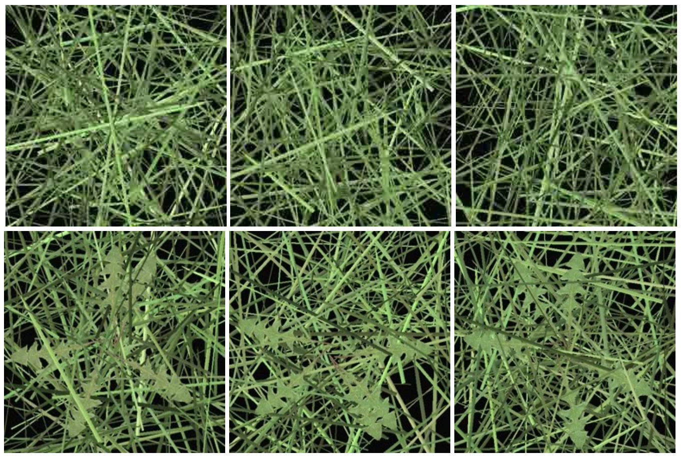

Figure 2: Examples of Synthetic Images
Top Row: contains no weeds, Bottom Row: contains weeds

each network architecture explored, each with a different seed used for generating random weights.

5. Naturally Captured Images

We captured video while scanning a lawn containing a variety of weeds and grasses, using an iPhone 6 Plus mounted on a robotic rover. The video was decoded and converted into a sequence of RGB images, and the individual images were manually classified into two groups:

Group 1: Grass only

Group 2: Grass and weeds, or weeds only

Among thousands of RGB images captured we selected a set of 100 images, 50 representing Group 1, and 50 representing Group 2.

We applied these 100 images to several DCNNs trained using synthetic data only; networks were selected among those which achieved 100% test accuracy using synthetic data.

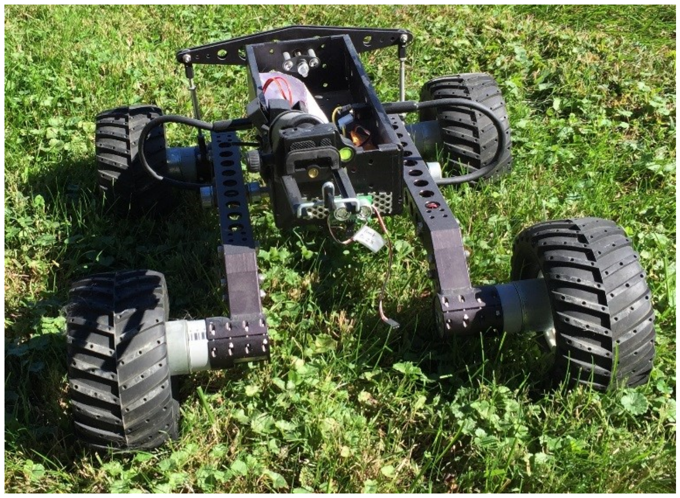

Figure 3: Robotic Rover Used to Capture Video for Testing

III. Results

1. Synthetic Data

Results based on training and testing with purely synthetic data are given in Table 3. For each set of DCNN hyperparameters we ran hundreds or thousands of independent training operations. Each training run used a different pseudo-random initialization of weights.

For the particular case of hyperparameters listed in Table 2 we ran 16,000 independent training operations and we found a large spread of convergence behaviors, as illustrated in Figure 5.

Table 2: Hyperparameters For High Volume Training Experiments

Convolutional layers 1-5	4 features
Fully connected layer 6	4 neurons
Fully connected layer 7	16 neurons

We estimated probability density function (PDF) of the random variable T_55_75, which we define as the number of epochs between the first time that a test accuracy of 55% is achieved, until the first time that a test accuracy of 75% is achieved.

The normalized histogram of T_55_75 is shown in the blue trace, in Figure 4. The shape of the PDF clearly resembles a Gamma distribution. For comparison, we plotted in red, the equation

$$y = \frac{1}{480}(x+2)^2 \, e^{-\frac{x+2}{6}}$$

We were able to obtain 100% accuracy on our test set of 1000 images using as few as 4 features per convolutional layer. Each training run involved application of 100 epochs of data, with each epoch consisting of 8000 images. It took hundreds of such training runs to produce one network with 100% accuracy on test data. As can be gleaned from Table 3, the probability of attaining 100% test accuracy increased as the number of features was increased.

A Linux workstation was used for network training, and roughly 100% of one core was consumed by the Caffe application. The workstation was based on Intel Xeon E5-2620 v3 @ 2.40GHz, 32 GB RAM, 2 CPUs, 12 cores total, RHEL 6 OS. The amount of time to complete 100 training epochs varied from 55 minutes for the smallest network to 11 hours for larger networks. The rightmost column in Table 3 represents the average time to attain 100% accuracy on test data, based on the fraction of runs that achieved 100% test accuracy and the time to complete one run. Of the network architectures tested, the most efficient configuration, from the standpoint of time to attain 100% test accuracy, had 16 features per convolutional layer and 128 neurons in fully connected layers 6 and 7. This architecture required an average of 4.29 hours to achieve 100% accuracy on test data.

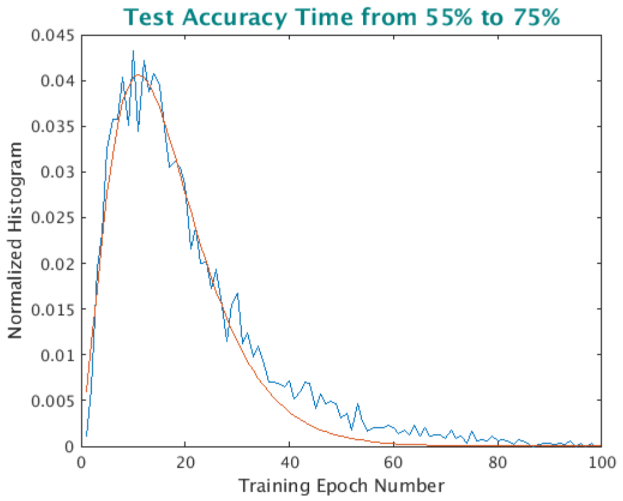

Figure 4: Time Difference From 55% to 75% Test Accuracy

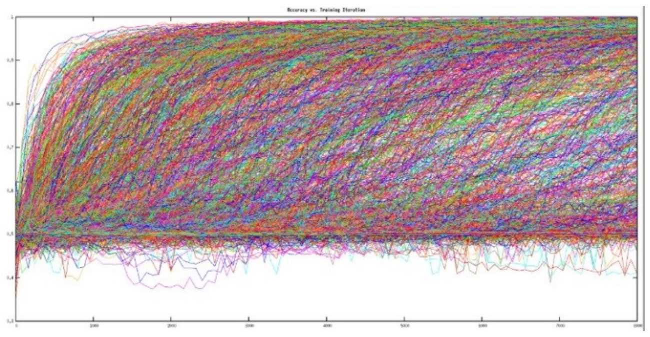

Figure 5: Plot of Accuracy vs. Training Iteration (overlaid results from 16,000 independent runs)

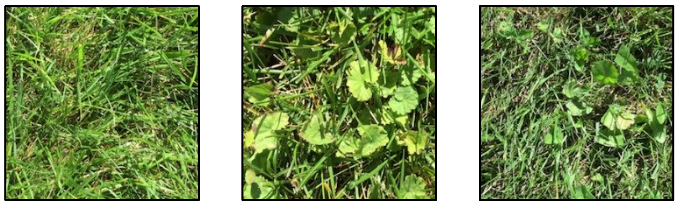

Figure 6: Natural Imagery Used For Testing
Left - correct H0, Center - Correct H1, Right - H1 misclassified H0

Table 3: Summary of Results of DCNN Training

Convo. Layers # Features	Fully Connected 6 # Features	Fully Connected 7 # Features	Percentage Achieving Test Accuracy Level			Time to Process 8000 Batches (800K images)	Average Time to Attain 100% Accuracy (hours)
			>=90	>=99	**100**		
4	4	16	30.1	8.3	0	-	-
4	4	256	35	10	0	-	-
4	8	256	56	33.3	0.629	0:55	145.73
4	16	256	83	61	0	-	-
4	128	128	80	56	0	-	-
8	8	64	41.6	35.6	2.2	1:31	
8	16	256	71.8	56.8	2.54	1:26	56.43
8	128	128	99	94	14	1:45	12.50
16	16	16	80	79	13	2:17	17.56
16	32	32	99	99	42	2:16	5.40
16	128	128	100	100	54	2:19	**4.29**
32	128	128	100	100	74	4:42	6.35
64	128	128	100	100	82	11:04	13.50

2. Naturally Captured Images

We found a correct classification rate of 95% (5% error) over the set of hand selected natural images. All five of the errors involved misclassifying images that actually represented H1 as representing H0. Figure 6 shows three of the natural images used for testing. The left-most image was correctly classified as representing H0. The center image was correctly classified as representing H1. The right-most image was incorrectly classified as representing H0.

IV. Conclusions

It appears that DCNNs can be a viable tool for practical applications of computer vision. We found that even very small DCNNs can handle multiple occlusions robustly and provide high accuracy image classification. Most notably we found that it is feasible and advantageous to train a DCNN using solely synthetic labeled data, even when the ultimate goal is to classify natural images. We studied the training behavior of DCNNs intensively, and found that the convergence time PDF resembles a Gamma distribution.

References

[1] Young, Stephen L., and Francis J. Pierce, eds. Automation: The Future of Weed Control in Cropping Systems. Springer, 2014.

[2] Ahmad, U., N. Kondo, S. Arima, M. Monta, and K. Mohri. "Weed detection in lawn field using machine vision: Utilization of textural features in segmented area." Journal of the Japanese Society of Agricultural Machinery (Japan) (1999).

[3] Tang, Lie, L. Tian, and Brian L. Steward. "Classification of broadleaf and grass weeds using Gabor wavelets and an artificial neural network." Transactions of the ASAE 46, no. 4 (2003): 1247.

[4] Watchareeruetai, Ukrit, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi. "Computer vision based methods for detecting weeds in lawns." Machine Vision and Applications 17, no. 5 (2006): 287-296.

[5] Watchareeruetai, Ukrit, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi. "Modified lawn weed detection: utilization of edge-color based SVM and grass-model based blob inspection filterbank." In International Conference on Neural Information Processing, pp. 30-39. Springer Berlin Heidelberg, 2007.

[6] Ahmed, Faisal, ASM Hossain Bari, A. S. M. Shihavuddin, Hawlader Abdullah Al-Mamun, and Paul Kwan. "A study on local binary pattern for automated weed classification using template matching and support vector machine." In Computational Intelligence and Informatics (CINTI), 2011 IEEE 12th International Symposium on, pp. 329-334. IEEE, 2011.

[7] Lee, Sue Han, Chee Seng Chan, Paul Wilkin, and Paolo Remagnino. "Deep-Plant: Plant Identification with convolutional neural networks." In Image Processing (ICIP), 2015 IEEE International Conference on, pp. 452-456. IEEE, 2015.

[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[9] Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding." arXiv preprint arXiv:1408.5093 (2014).

[10] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In Aistats, vol. 9, pp. 249-256. 2010.