An Effective Segmentation Algorithm of Apple Watercore Disease Region Using Fully Convolutional Neural Networks

Zeng PENG, Cheng CAI*
1. College of Information Engineering, Northwest A&F University, Yangling,712100.
E-mail: chengcai@nwsuaf.edu.cn

***Abstract*—The apple watercore disease region detection is crucial to apple disease analysis in life science. However, traditional methods cannot be applied to this situation, because the watercore region varies gradually according to the apple region, and lighting variations and the bruise also give many false alarms. Now there is no such a stable and accurate method to cope with this difficulty. In this paper, we propose an effective apple watercore disease region segmentation scheme, which is intended to extract comprehensive representations of image on the basis of multilayer convolutional neural networks. With the learnt hierarchical feature representations from millions of pixels, false alarms caused by lighting variation and bruise are also restrained. Experiments based on a large self-designed dataset of apple watercore disease images show encouraging results.**

I. INTRODUCTION

The promise of Deep ConvNets is to discover and present abstract hierarchical models that express probability distribution in dealing with computer vision tasks, and has enjoyed fantastic results recently in the field like classification[1],[2], bounding box detection[3],[4],[5], natural language processing[6],[7],[8], machine translation[9],[10],[11], speech recognition[12],[13], image generation[14],[15], and semantic segmentation[16],[17]. All these fabulous successes are primarily based on the process of backpropagation and fine-tuning. The accuracy of new approaches improves year by year owing to the deeper networks.

Apple watercore disease region detection is of great importance in life science research, which greatly influences apple quality and thus affects its storage and transportation. As apple flesh in watercore region is darker than that of non-watercore region, it is often easier to recognize dark-colored watercore regions. The real difficulty lies in those light-colored watercore regions, whose color is affected by bruise and lighting conditions. To solve this problem, Tao Dai and Cheng Cai[18] used the Bayesian classifier with the histogram technique to segment watercore regions, which achieved good performance in specific status. But without considering lighting variations and bruise, traditional machine learning methods like the Bayesian classifier are impractical to effect in this circumstance.

Semantic segmentation is a key problem in computer vision, and the traditional ConvNets is often used to segment in prior research. To classify one pixel, an image patch around the pixel is used as the CNN input for training and prediction. This method has several shortcomings. First, storage costs are rather high. For example, the size of the image patch used for each pixel is 15×15. Then the filter window continuously slides one after another to discriminate the CNN. Therefore, the required storage space increases sharply according to the number and size of sliding windows.

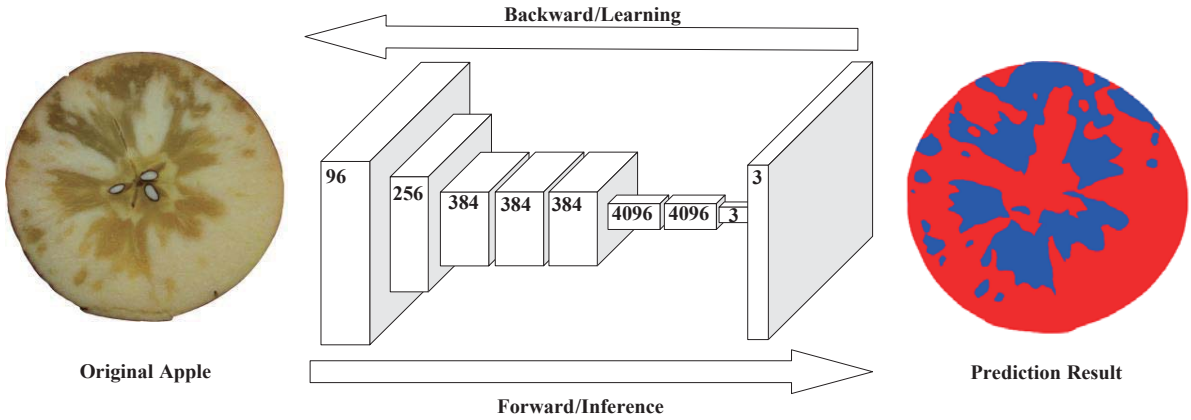

Fig. 1. Dense predictions for per-pixel tasks

Second, the calculation is of low efficiency. The adjacent pixel patches are essentially repetitive, and the convolution is calculated one by one for each pixel patch, and this calculation is also repeated to a large extent. Third, the size of pixel patches limits the size of the receptive field. Usually pixel patches are much smaller than the whole image. As only some local features can be extracted, the performance of the classification is limited.

However, the convolutionalized CNN, called the Fully Convolutional Networks(FCN) in our study, can solve this kind of dense labeling problem fundamentally, for it analyzes the pixel category from its abstract features, thus extending the classification from image level to pixel level. This method is computationally efficient with no need for pre- and post-processing, and helps to get dense outputs from arbitrary-sized inputs. In the process of FCN, transposed convolution is applied to upsampling the low-resolution features learnt by the preceding convolutional architectures. In this paper, we will make the very first application of the Fully Convolutional Networks[16] to apple watercore disease region segmentation. We are about to try an end-to-end, and pixel-to-pixel Networks to get the state-of-the-art results on the apple watercore dataset.

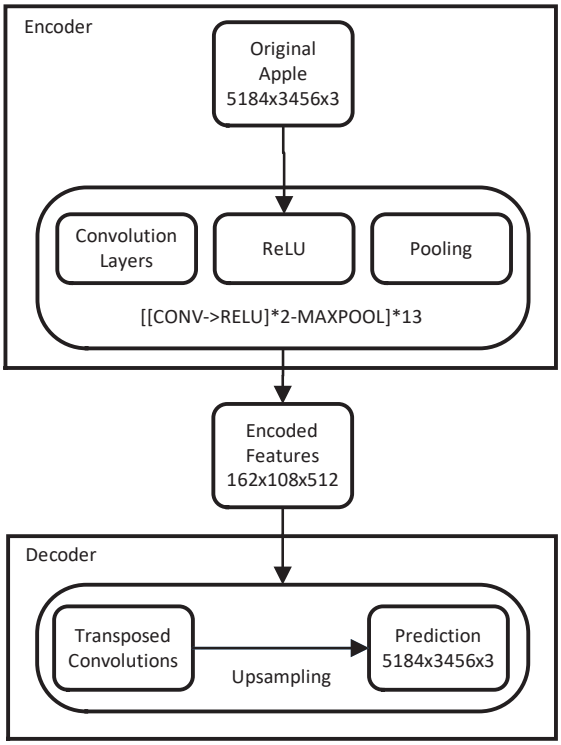

Fig. 2. Fully Convolutional Networks architecture

In the following sections, we will first elaborate on how convolutional architectures, as an encoder, extract high dimensional features in Section 2. Then in Section 3, we will demonstrate how transposed convolution, as a decoder, decodes the features, and then apply this Networks to apple watercore region segmentation. In Section 4, we will introduce traditional machine learning methods to color pixel classification. In Section 5, we will analyze our experiment results in comparison to some traditional machine learning methods. Finally, a conclusion will be made briefly in Section 6.

II. Encoder: Feature Extraction Using Convolutional Architectures

When we are doing linear regression or logistic regression, intrinsically we are doing data mapping. We can either map it to one or more discreate labels or to continuous space. Generally, for some uncomplicated data, it is easy for us to do linear variation to obtain the fitting function. But when encountered with a complicated problem involving high dimension, as in our research, the data are usually linearly inseparable. Therefore, we turn to convolutional architectures, in which the ConvNets can easily extract features so as to obtain better filters for specific problems through backpropagation.

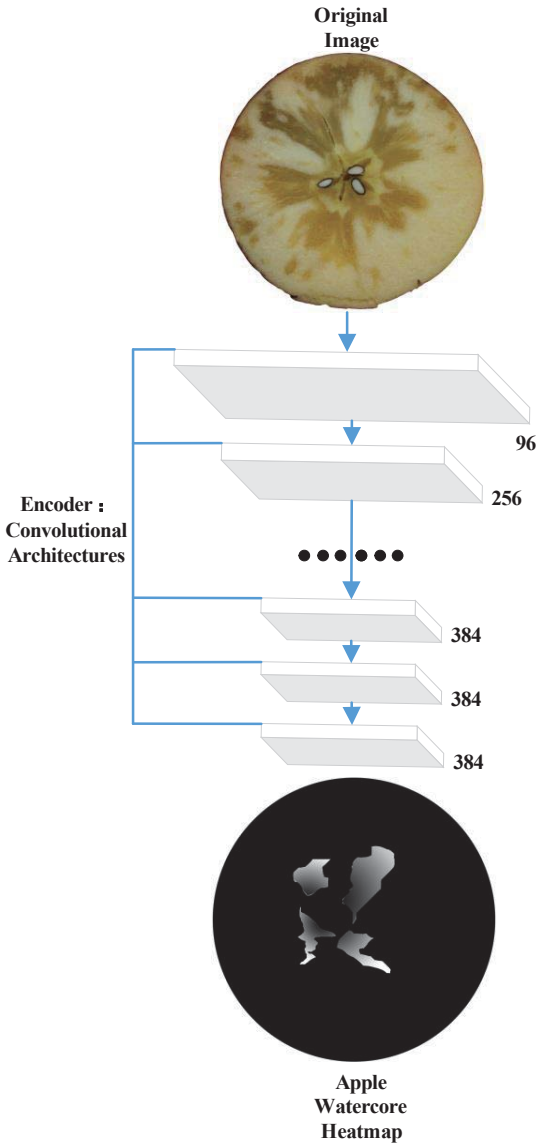

Fig. 3. Encoder: Convolutional Architectures

A. Convolution layers

The convolutional layer is the core of the whole architecture, which also consumes most of the computational resources. The parameters of ConvNets layers are made up with learnable filters, each of which is rather small spatially with the same depth of input data. Every unit is made to connect to a region patch of the whole input data, and the connected space is called receptive field.

Suppose our data size is $H_1 \times W_1 \times 3$(3 means the color channel), and then the size of the filter is $3 \times 3 \times 3$. The hyper-parameters include the number of the filters K, the spatial size of the filter F, the stride S, and the zero-padding P which controls the size of the data volume. In forward propagation, filter windows slide through the width and the height, doing dot products, and the step is given by the stride. That is, the window slides S pixels every time. Then we can get the size of output data volume after one time convolution:

$$Output\ size = W_2 \times H_2 \times D_2 \tag{1}$$

Certainly we can infer:

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1 \tag{2}$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1 \tag{3}$$

$$D_2 = K \tag{4}$$

B. Activation functions

The activation functions are aimed at adding non-linear factors to solve the insoluble problem by using linear models. Good activation functions usually have some inherent characteristics. First, they need to be differentiable so as to optimize the parameter by gradients. Second, the monotonicity of the function is often required to obtain a convex function. When the outputs of the activation function are limited, the optimization method based on gradients will be more stable. These are why we use activation functions.

There are a number of activation functions, such as Sigmoid, Tanh, and ReLU. All of them have some shortcomings. Sigmoid kills the gradients in saturation, and the outputs are not zero-centered, which affects the gradient descent. Tanh has the same saturate problem, but it is zero-centered. In this case we choose ReLU as our activation function, and the expression is:

$$f(x) = \max(0, x) \tag{5}$$

Compared with Sigmoid and Tanh, ReLU can sharply accelerate the process of gradient descent, requiring less computation. The drawbacks emerge when a big gradient flow through the ReLU unit. After the update, this unit will no longer activate any data, and then its gradient will become zero. But we can set an appropriate and small learning rate to reduce the dead units.

C. Pooling layers

It is often the case that we insert pooling layers to the successive convolution layers regularly to reduce the spatial size of the data volume. With this operation, we can reduce the number of the parameters in the Networks so that we can save more computation resources, free from the restrictions of the overfitting. We use the pooling filters for down-sampling, whose size is 2x2, and the stride is 2. The strategy of max pooling is taking a max over 4 numbers (in 2x2 region).

III. Decoder: Feature Resolution Using Transposed Convolutions

A. Encoded features

So far, we have completed the encoding procedure where data features are extracted through convolution architectures. In the visualizations of the feature map after optimization, the learnt features of the convolutional architectures differ greatly from human semantic features, for they are highly abstract and incomprehensible. But they still contain sufficient information for subsequent task-specific decoders to tackle different problems. Therefore, we apply the Transposed convolution as a semantic segmentation decoder to resolve the learnt features.

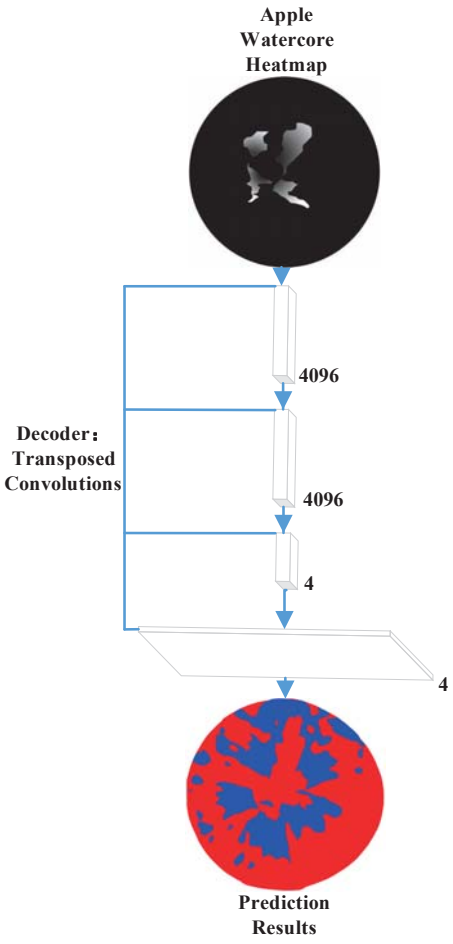

Fig. 4. Decoder: Transposed Convolutions

B. *Transposed convolution*

The whole process of transposed convolution is regarded as up-sampling. In the traditional Convolutional Neural Networks, we choose a couple of fully connected layers after convolution architectures for classification. In completing the segmentation task, we do transformation on fully connected layers, called convolutionalization, more specifically. As seen in Fig. 4, the size of the input data is 5184×3456×3. After flowing through the convolution architectures, down-sampled by pooling layers, the size becomes 162×108×512. Then in fully connected layers, the units are 4096. Later a filter is used, whose receptive field is 162×108, stride is 1, and the number is 4096. Fig. 5 illustrates the process of convolutionalization in detail, left part is the traditional CNN for classification, the right part is the new architecture of FCN. We changed the probability results of classification into output images, and that is what we call semantic segmentation.

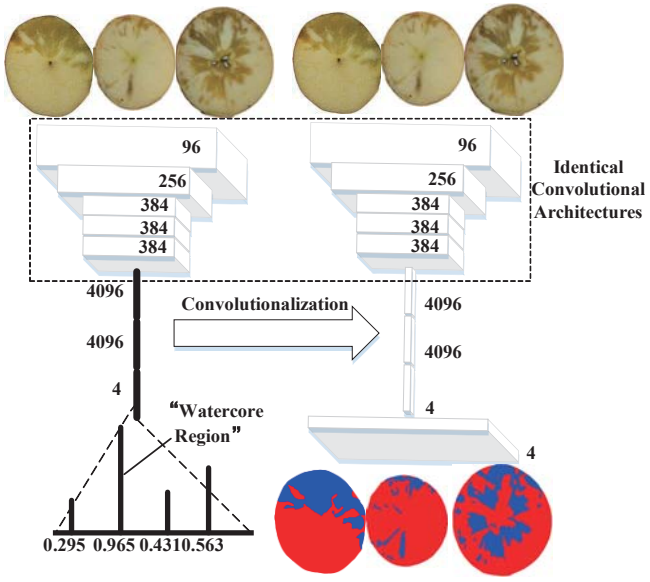

Fig. 5. Convolutionalization

C. *Prediction*

Up to now we have illustrated the structure of the whole Neural Networks. With the convenient of an end-to-end Networks, we can receive arbitrary-sized data without any pre-operation. Like other traditional methods of semantic segmentation, the Convolutional Neural Networks [19], [20], [21], [22] uses patch-wise training, and performs the forward flow of inference in the whole image data at a time, which sharply decreases the computation and combines both global information and local information in hierarchical architectures. In our training, however, uniformed data, after getting input, are allowed to flow through the whole Networks for inference and backpropagation, thus minimizing our loss function. In the end, we obtain our learnt parameters. In the next section, we will describe the performance of our learnt architectures.

IV. TRADITIONAL COLOR PIXEL CLASSIFICATION

Previously, Son Lam Phung *et al*[23] used the Naïve Bayesian classifier with a histogram technique for color pixel classification. The Bayesian decision rule is based on probability and statistical theory with solid mathematical basis in statistical pattern classification. Primarily the Bayes theorem is

$$P(B \mid A) = \frac{P(A|B)P(B)}{P(A)} \tag{6}$$

After some ordinary transformations, we consider one pixel *Ci* as a watercore pixel if

$$\frac{P(W|C)}{P(NW|C)} = \frac{P(C|W)}{P(C|NW)} = \frac{\prod_{i=1}^{n} P(Ci|W)}{\prod_{i=1}^{n} P(Ci|NW)} > \tau \tag{7}$$

Where *W* represents watercore pixel, *NW* represents apple flesh pixel, and τ is the threshold obtained in training. To get the optimal τ and prior probability, *pdfP* and *pdfN* are required. *pdfP* is a 255×255×255 tensor, which saves the number of occurrences of watercore pixels in whole RGB space. *pdfN* is also a tensor with the same size that saves the number of occurrences of apple flesh pixels. Then we define

$$\frac{\prod_{i=1}^{n} P(Ci|W)}{\prod_{i=1}^{n} P(Ci|NW)} = cost\ (r,g,b) \tag{8}$$

$$cost(r,g,b) = \frac{pdfP(r,g,b) \Big/ \sum_{i=1}^{n} pdfP(r,g,b)}{pdfN(r,g,b) \Big/ \sum_{i=1}^{n} pdfN(r,g,b)} \tag{9}$$

After training we get one tensor *cost* with the size of 255×255×255, which saves the whole cost value of each color. So we can make predictions on our validation sets at each pixel color. If

$$cost(r,g,b) > \tau \tag{10}$$

is true, the color with RGB value *r,g,b* is a watercore region pixel.

V. DATA MATERIAL AND EXPERIMENTAL RESULTS

A. *Apple watercore diease region dataset*

This dataset is designed and implemented by the College of Life Science, Northwest A&F University, Shaanxi, China. We collected 546 apple slices from altogether 182 high definition images with the resolution of 5184 × 3456 in different complicated light conditions. It is intractable to recognize the connection areas between watercore regions and apple flesh regions along with the bruise and lights. Each image contains 17,915,904 pixels, and 3,296,526,336(about 3.3 billion) pixels in total.

Fig. 6. Original Data

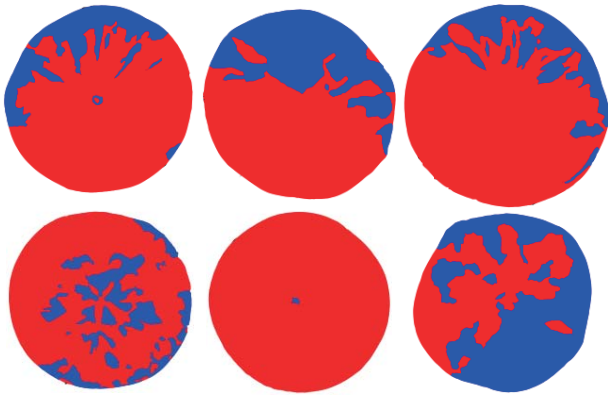

Fig. 7. Ground Truth

TABLE I
APPLE WATERCORE DISEASE DATASET FOR TRAINING AND TESTING.

Dataset	Number of Images.	Apple Flesh Pixels	Watercore Pixels
Training	410	773.5 million	273.7 million
Testing	136	223.9 million	137.8 million

Then different areas of each image have been marked manually as the ground truth. Respectively, blue indicates watercore region, and red indicates apple flesh. The labeled data have the same resolution as to original data. The whole dataset is available at College of Information Engineering, Northwest A&F University, Shaanxi, China.

B. Performance evaluation results

In our experiments, we used quantitative methods to evaluate the performance of segmented results, which possesses high objectivity and repeatability. Three standard performance measures are adopted: mean accuracy, per pixel accuracy, and mean Intersection over Union (Mean IoU for short)[24] as follows.

Pixel accuracy:

$$\sum_i Num_{ii} \Big/ \sum_i Total_i \tag{11}$$

Mean accuracy:

$$(1/Num_{cl}) \sum_i Num_{ii} \Big/ Total_i \tag{12}$$

Mean IoU:

$$1/Num_{cl} \sum_i Num_{ii} \Big/ (Total_i + \sum_j Num_{ji} - Num_{ii}) \tag{13}$$

Where Num_{ij} is the number of pixels belong to class i that classified as class j, Num_{cl} is the total number of classes, and $total_i$ is the total number of pixels of class i, which equals to the summation of Num_{ij}. Our parameter update method is SGD with momentum. In our experiment, the learning rate is 10^{-4} while the momentum is 0.9. The receptive field of filter is set as 3×3, and the convolution stride is set as 1. We choose the Max-pooling strategy with 2×2 pixel window, with the stride at 2. Then we zero-initialize the layer of scoring, by adopting the dropout in convolutional architectures. We fine-tune all the parameters through backpropagation and parameter search, which takes about one day on a single GPU. All models are implemented and tested with TensorFlow[25] on Nvidia GTX 1070.

As it is shown in Table II, extraordinary improvements have been achieved by the deep architecture method FCN. It is found that the hierarchical layers of convolution architectures can learn a remarkable and representative feature that takes different lighting condition and bruise into account, compared with the Bayesian Classifier.

TABLE II
SEGMENTATION ACCURACY ON THE APPLE WATERCORE DATASET.

	Mean acc.	Pixel acc.	Mean IoU
FCN	**75.9**	**87.5**	**73.7**
Bayesian	67.3	72.9	57.8

We present all our experimental results in Fig. 8. Obviously there is significant difference between the Bayesian Classifier and FCN whose performance is clearer and more persuasive. We choose these results to testify the outstanding performance made by FCN with hierarchical architectures.

The rows (a) are the original apple slices selected from our dataset. The rows (b) are the ground truth. The rows (c) are the output images chosen from the Bayesian Classifier. The rows (d) are the output results from FCN. We used the blue color for correct hits, green for missing pixels, and red for false positives.

As you know, a traditional machine learning method like Bayesian classifier lacks representative ability especially when

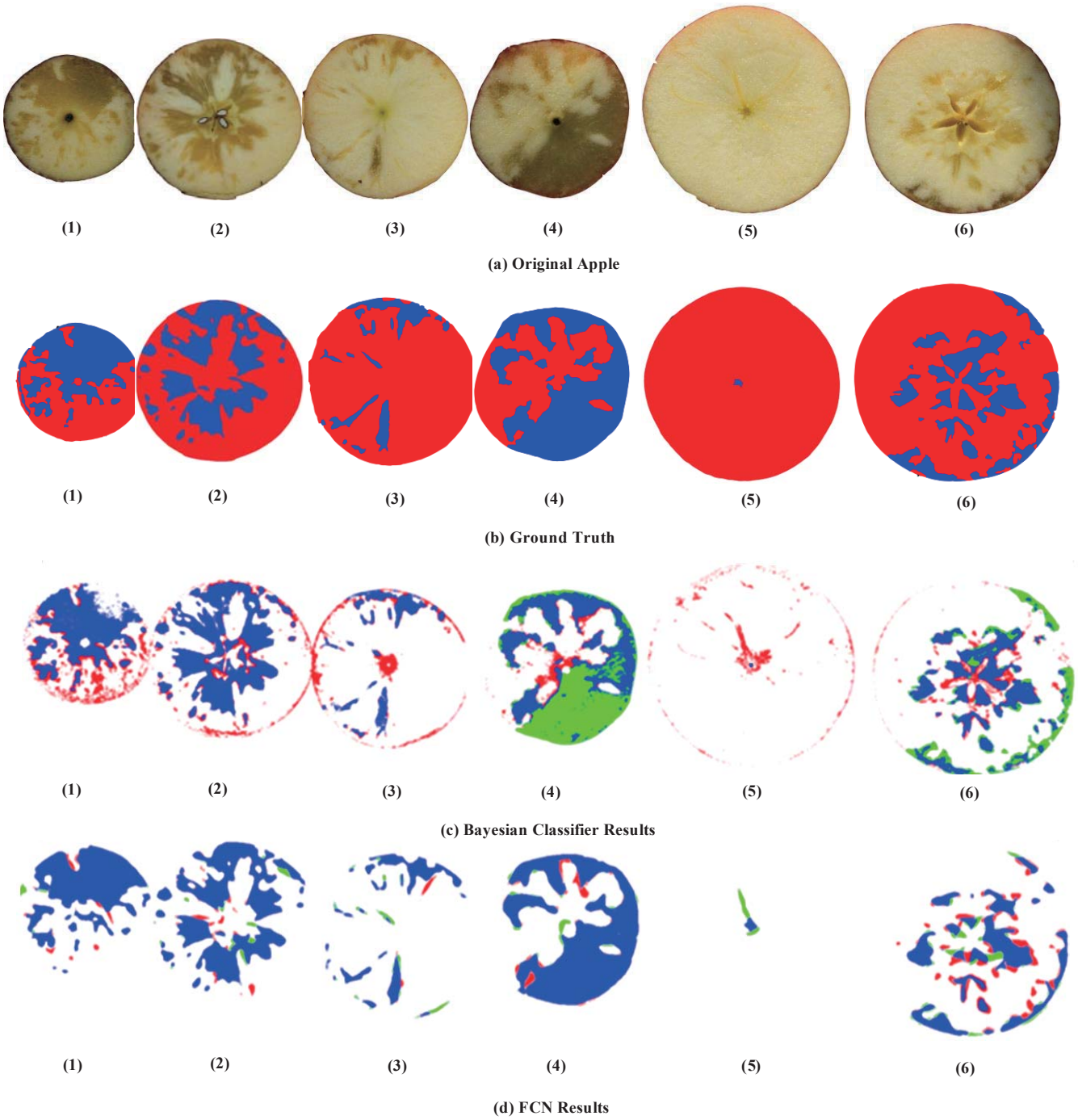

Fig. 8. A thorough results comparison between Bayesian classifier and FCN. (a-1) to (a-6) input original apple image. (b-1) to (b-6) ground truth. (c-1) to (c-6) Bayesian classifier results. (d-1) to (d-6) results from FCN.

the lighting condition is unstable. As in the apple (c-4) with the Bayesian classifier, a large green area means that a lot of watercore pixels are ignored, while those pixels in the apple (d-4) with FCN are not, the watercore pixels are perfectly segmented. It is because the light conditions affect the Bayesian classifier severely, but the FCN still holds the representability.

You can also check the apple (c-1), (c-2), (c-3), and (c-5) with the Bayesian classifier row. The apple edge is considered as watercore region by error, but it is only bruise, not watercore region. Compared with the apples in FCN method where the edge does not be segmented at all. This is a clear and straightforward evidence of FCN's outstanding performance.

In addition to the above-mentioned evaluation methods, we also evaluate our algorithm and results using ROC curve. We sort the samples results with our two classifiers, and predict the sample as positive by following the same procedure. After computing two values, we get the ROC curve.

Then we define the True positive rate and false positive rate as follows:

$$\mathrm{TPR} = {}^{TP}/_{(TP + FN)} \tag{14}$$

$$\mathrm{FPR} = {}^{FP}/_{(TN + FP)} \tag{15}$$

TABLE III
SEGMENTATION STATISTICAL DATA ON THE APPLE WATERCORE DATASET.

Pixel data	True Positive	True Negative	False Positive	False Negative
FCN	**234.5 million**	**634.2 million**	134.1 million	34.7 million
Bayesian Classifier	134.4 million	434.8 million	**345.8 million**	**129.5 million**

Therefore, the ROC curve can be used to evaluate the generalization performance of the algorithm with different tasks. Very often the larger area the curve encloses, the better performance the classifier achieves.

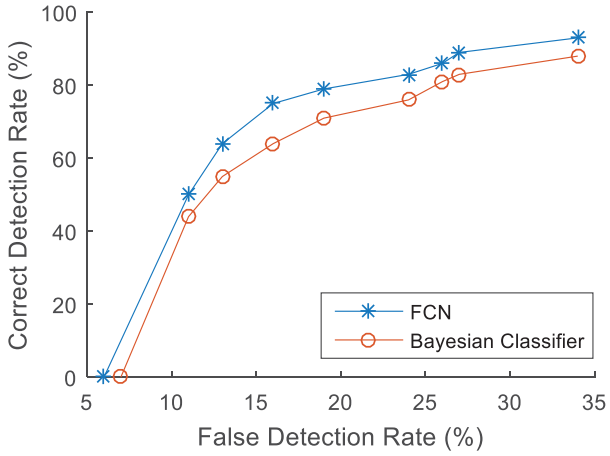

Fig. 9. ROC curves of apple watercore region segmentation.

The ROC curves clearly indicate that FCN performs better than the Bayesian Classifier, which in turn confirms that the hierarchical abstract models can extract precise features and obtain outstanding results after training.

VI. CONCLUSIONS

In this paper, we propose a novel approach by applying the Fully Convolutional Neural Networks to semantic segmentation, which has achieved state-of-art results compared with traditional machine learning methods. The evaluation results show that the deep layered architectures perform extremely well in representing high dimensional features, particularly in our dataset, and varied lighting conditions and bruise features are beautifully learnt and represented. However, nothing is perfect. Results from the Networks are not exquisite. Even though the 8 times of up-sampling is better than 32 times, the output images are indistinct and smooth, lacking sensitive details. Also this method only classifies pixels themselves without considering their interconnections, and ignores spatial regularization, thus having unsatisfactory spatial consistency.

To prepare for future research work, so far we have designed and implemented a brand-new dataset on the apple watercore disease region detection.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (Project 61202188), National Undergraduate Scientific and Technological Innovation Project (Project 201710712060), China. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-scale Image Recognition," *International Conference on Learning Representations*, pp. 196- 201, 2015.

[3] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440-1448, 2015.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.

[5] C. Szegedy, A. Toshev and D. Erhan, "Deep Neural Networks for Object Detection," in *Advances in Neural Information Processing Systems*, pp. 2553-2561, 2013.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality," in *Advances in Neural Information Processing Systems*, pp. 3111-3119, 2013.

[7] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic Memory Networks for Natural Language Processing," *Computing Research Repository, abs/1506.07285,* 2015.

[8] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint Learning of Words and Meaning Representations for Open-Text Semantic Parsing," in *Artificial Intelligence and Statistics*, pp. 127-135, 2012.

[9] M. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural

machine translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 11-19, 2015.

[10] R. Sennrich, B. Haddow and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," *Annual Meeting of the Association for Computational Linguistics*, pp. 210-221, 2016.

[11] M. Luong, H. Pham and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation,", *Empirical Methods in Natural Language Processing*, pp. 109-114, 2015.

[12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Banjouke, P. Nguyen, and T. N. Sainath, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine,* vol. 29, pp. 82-97, 2012.

[13] A. Graves, A. Mohamed and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* pp. 6645-6649, 2013.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in neural Information Processing Systems*, pp. 2672-2680, 2014.

[15] A. Radford, L. Metz and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *International Conference on Learning Representations*, pp. 134-145, 2015.

[16] J. Long, E. Shelhamer and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015.

[17] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, " DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," in *International Conference on Learning Representations,* pp. 431-440, 2014.

[18] T. Dai, C. Cai, H. Ma, and S. Li, "Apple Watercore Region Detection Using Color Pixel Classification," *ICIC Express Letters,* vol. 9, pp. 1213-1219, 2015.

[19] P. Pinheiro and R. Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling," in *International Conference on Machine Learning*, pp. 82-90, 2014.

[20] Y. Ganin and V. Lempitsky, "N^4-fields: Neural network nearest neighbor fields for image transforms," in *Asian Conference on Computer Vision*, pp. 536-551, 2014.

[21] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, " Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, pp. 1915-1929, 2013.

[22] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, " Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," in *Advances in Neural Information Processing Systems*, pp. 2843-2851, 2012.

[23] S. L. Phung, A. Bouzerdoum and D. Chai, "Skin Segmentation Using Color Pixel Classification: Analysis and Comparison," *IEEE transactions on pattern analysis and machine intelligence,* vol. 27, pp. 148-154, 2005.

[24] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, " The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision,* vol. 111, pp. 98-136, 2015.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin, " TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467,* 2016.