Multi-Scale Boundary Detection via Mixture Convolution-Deconvolution Network

Ga Wu
Oregon State University
Corvallis, OR 97331
wug2@oregonstate.org

Xu Xu
Oregon State University
Corvallis, OR 97331
xuxu@oregonstate.org

Abstract

The purpose of this project is to extend convolution-deconvolution network to support multiple convolution-deconvolution network configuration in a single network. In previous works, people carefully adjusted the number of layers in the network to improve the performance. However, the network is task sensitive, which cannot easily be applied to other related tasks without tuning the networks's architecture such as changing the number of hidden units in each layer and adding or removing layers. Image boundary detection problem is one of the typical problems that require to detect boundaries in different level of details, in which deeper network can find larger bolded boundaries but also sacrifices useful information to find small thin boundaries. This raises an interesting question: how to combine multiple configurations to make robust prediction? In this work, we combine multiple convolution-deconvolution configurations in an single network which automatically adjusts the weights of multiple structures to make reasonable prediction. Due to time limitation, the implementation is not fully complete. But, we successfully extended Keras to support convolution-deconvolution structure, which was empty in both Theano and Keras implementation.

1. Introduction

Edge detection as a fundamental problem in Computer vision field has been intensively studied for years. Through detecting sharp changes in image brightness, we can collect set of connected curves that indicate boundaries of objects. Applying an edge detection algorithm to an image can significantly reduce data requirement, because the subsequent task of interpreting the information in the original image is substantially simplified.

There has been a very long history of computational edge detection. In the early stage of this area,edge detectors like Sobel detector[12],Canny detector[4] mainly aim at computing local gradients to detect edges.Later, edge detection accuracies are improved through careful manual design in information theory based approaches , such as Statistical Edges[13],gPb[1], as well as by learning-based approaches like Multi-scale[17], Structured edges[6]. In addition, there has been a trend in using Convolutional Neural Networks (CNN)[8][14]to automatically learn hierarchical feature[20]. As suggested in[17], multi-scale can help boundary detection by combining the strengths from both large-scale detection(robust but poor localization) and small-sclae detection(detail-preserving but sensitive to clutter). So naturally, we would like to use CNN to train multi-scale filters on multiple layers for edge detection, thus exploiting these information for edge detection.

The hard coded optimal filters has their drawbacks that require careful adjustments and may not be applicable to some specific type of images. Therefore, learning filters from data becomes the optimal solution.

Convolutional neural networks in computer vision field are very popular in various recognition problems such as image classification[14][18], semantic segmentation[16][7], as well as object detection[9][10]. The powerful feature learning ability of CNN leads to great performance of prediction even with simple classifier.

In a recent study, people began to use convolution-deconvolution structure to solve structured pixel-wise labeling problems, and deconvolution network is introduced in [21] to reconstruct input images, however most of these related works naively reconstruct the image size output by performing single step deconvolution. One of the most influential related work is learning semantic segmentation through deconvolution network[15]. This approach is also employed to visualize activated features in a traind CNN to update network architecture to improve performance.

However, as a practical problem, the number of layers in deconvolution network requires careful tuning and lots of experiments to justify its reliability. This raises an interesting question: can we consider multiple possible structures all together and let data tell us which to use?

There are several implementation challenges in this project: 1) there is no existing code in Theano and Keras that implement max unpooling function, and the max pooling function does not provide index information after giving output. 2)BSDS500 data was in matlab format, we need to convert such data set into a proper format that can be read by Python.

In this work, we want to 1) generate useful toolbox to handle convolution-deconvolution structure in Keras and Theano environment. 2) prove our initial idea of combining multiple deconvolution structures to do reliable prediction.

The report will be delivered in the following structure. In section 2, we describe the proper convolution deconvolution structure for making boundary detection. In section 3, we give detailed description of combining multiple deconvolution structures in a single neural network. In section 4, we introduce some important functions in our implementation of deconvolution network. In section 5, we list out our experimental results and make analysis. The conclusion will be in the last section as usual.

2. Network Description

This section discusses the basic architecture of convolution-deconvolution network, and describes modified maxpooling and unpooling knowledge for deconvolution task.

2.1. Architecture

Table 1 describes the configuration of the traditional convolution-deconvolution network. First half of the network corresponds to feature extractor that maps images into compact vectors of features, whereas the rest of the network reconstructs proper shape of ground truth image that contains only boundaries and blank rest area.

Since the ground truth can be represented in 2 dimensional boolean type structure, We add an additional layer in the network to convert output in range of $[0, 1]$. To maintain the performance and reduce computational complexity, in each convolutional layer there are over 36 filters. This number is determined by considering the type of this task is boundary detection, in which larger number of filter doesnt help much.

2.2. Max Pooling and Unpooling

After two convolutional layers, there is a pooling layer. The pooling layer takes small region, usually 2x2 block, from the output of convolutional layers and subsamples it to produce single output from the block. Among the several candidate ways to do pooling task, max-pooling is more prefered in this task because of its nature to select activations that lose less information.

However, reversing a max pooling operation is much more difficult than that of other pooling method. We need not only store max pooling result, but also store index mask of max pooling. And, we recover original size of images by filling zeros into the place of inverse of the mask as shown in figure 1.

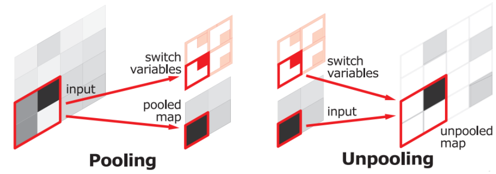

Figure 1. Max pooling and unpooling: This figure was published on paper 'Learning Deconvolution Network for Semantic Segmentation'

name	kernel size	stride	pad	output size
input	-	-	-	$480 \times 320 \times 3$
conv1-1	3×3	1	1	$480 \times 320 \times 32$
conv1-2	3×3	1	1	$480 \times 320 \times 32$
pool1	2×2	2	0	$240 \times 160 \times 32$
conv2-1	3×3	1	1	$240 \times 160 \times 32$
conv2-2	3×3	1	1	$240 \times 160 \times 32$
pool2	2×2	2	0	$120 \times 80 \times 32$
conv3-1	3×3	1	1	$120 \times 80 \times 32$
conv3-2	3×3	1	1	$120 \times 80 \times 32$
pool3	2×2	2	0	$28 \times 28 \times 32$
conv4-1	3×3	1	1	$60 \times 40 \times 32$
conv4-2	3×3	1	1	$60 \times 40 \times 32$
pool4	2×2	2	0	$30 \times 20 \times 32$
conv5-1	3×3	1	1	$30 \times 20 \times 32$
conv5-2	3×3	1	1	$30 \times 20 \times 32$
conv5-3	3×3	1	1	$30 \times 20 \times 32$
pool5	2×2	2	0	$15 \times 10 \times 32$
fc6	15×10	1	0	$15 \times 10 \times 32$
fc7	15×10	1	0	$15 \times 10 \times 32$
deconv-fc6	15×10	1	0	$15 \times 10 \times 32$
unpool5	2×2	2	0	$30 \times 20 \times 32$
deconv5-1	3×3	1	1	$30 \times 20 \times 32$
deconv5-2	3×3	1	1	$30 \times 20 \times 32$
deconv5-3	3×3	1	1	$30 \times 20 \times 32$
unpool4	2×2	2	0	$60 \times 40 \times 32$
deconv4-1	3×3	1	1	$60 \times 40 \times 32$
deconv4-2	3×3	1	1	$60 \times 40 \times 512$
unpool3	2×2	2	0	$56 \times 56 \times 32$
deconv3-1	3×3	1	1	$120 \times 80 \times 32$
deconv3-2	3×3	1	1	$120 \times 80 \times 32$
unpool2	2×2	2	0	$240 \times 160 \times 32$
deconv2-1	3×3	1	1	$240 \times 160 \times 32$
deconv2-2	3×3	1	1	$240 \times 160 \times 32$
unpool1	2×2	2	0	$480 \times 320 \times 32$
deconv1-1	3×3	1	1	$480 \times 320 \times 32$
deconv1-2	3×3	1	1	$480 \times 320 \times 32$
output	1×1	1	1	$480 \times 320 \times 1$

Table 1. Configuration of traditional convolution-deconvolutional network for boundary detection. Activation function used in our experiment is ReLu. The only exception is the output layer, which uses sigmoid function to generate result in range of [0,1].

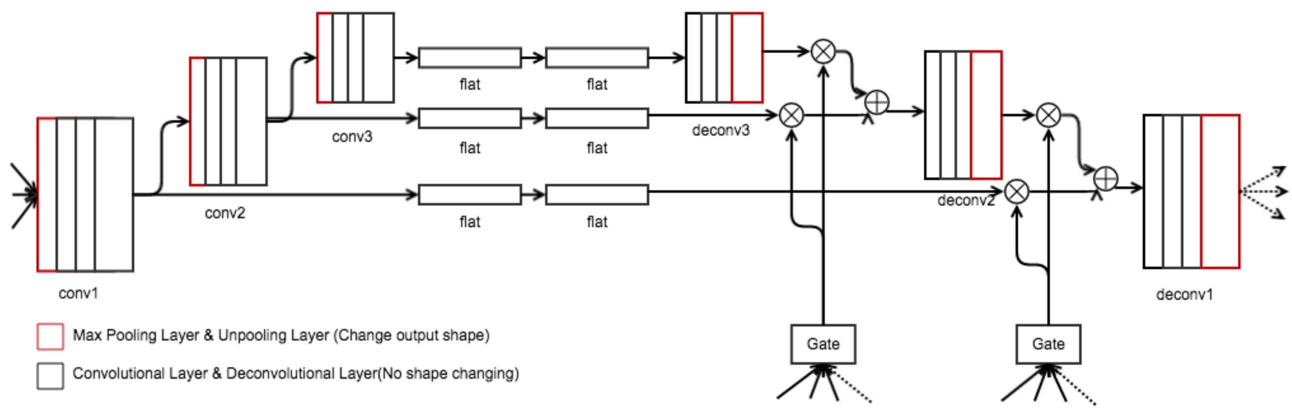

Figure 2. Combination of multiple convolution-deconvolutional configurations in Single Network.

2.3. Automatic Network Configuration

We notice that the traditional structure of convolution-deconvolutional network requires researchers to make great effort to configure the number of layers or number of filters to improve the network performance. Larger filter size would not hurt the performance when the data size is big enough, whereas more layers in the network may dramatically impact the performance because of losing information during the max pooling procedure. Therefore, we introduce a novel structure to allows the network to learn its configuration without human efforts.

Figure 2 shows the proposed structure of convolution-deconvolutional network. We use a gate to control the merging of information from previous level deconvolution and flattened inner representation.

The gate here is simply a sigmoid function, which accepts some constant value as input and it will provide a float type value in range of (0,1).

$$Gate(V) = \frac{1}{1 + \exp(-w_v^T V)} \tag{1}$$

Gate can be learned during training as the idea applied in LSTM[11] and Hightway Network[19].

$$D1.I = D2.O \times G + F1.O \times (1 - G) \tag{2}$$

Where D1,D2 represent deconvolutional structure 1 and 2, I,O indicate input and output, G is gate.

The difference between the gate in our model and that in LSTM RNN structure is whether to use input and output of the network as input of the gate. The value of gate of our model is constant for images that belong to same type of problem, which means the strategy of LSTM is unfeasible in our case. Therefore, we are looking for graphical model method to address weight adjusting.

So far, let us assume the value is already learned through some black technology.

3. Implementation Detail

Due to there is no existing code about deconvolution in Theano(also its extensions like Keras) and Max unpooling, we implemented convolution and deconvolutional network in Python to fill this blank. The code has complete convolution, deconvolution, maxpooling and reverse maxpooling layer classes.

Code is on GitHub[1].

We answered one interesting question that lots of people keep asking for the solution– How to efficiently reverse maxpooling that can be run on GPU. Our answer is simply and clean. We use the symbolic programming property of Theano to address this problem.

For obtain activation index of maxpooling, We only need to replace the gradient value to be 1 as shown below:

```
mask = T.grad(
None,
wrt=input,
known_grads={
pooled_out: T.ones_like(pooled_out)
    }
)
```

For computing reverse maxpooling, we do it again!

```
reversed = T.grad(
None,
wrt=mask,
known_grads={pooled_out: pooled_out}
)
```

Since the BSDS500 data is originally published in matlab format, we need to convert it into a format that python can

[1] https://github.com/wuga214/Boundary-Detection-via-Convolution-Deconvolution-Neural-Network-with-BMA.git

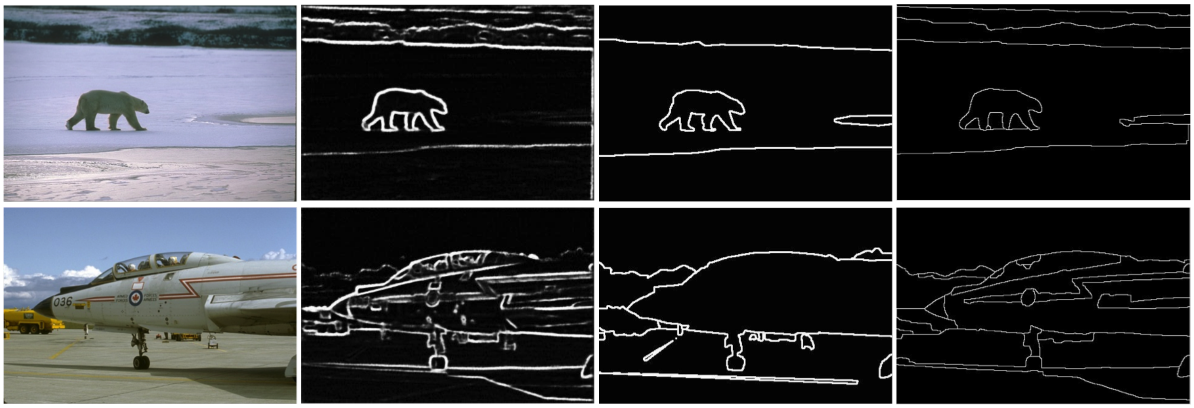

Figure 3. Edge Detection through Convolution Deconvolutional network. From left to right: Image, Detected Boundaries, Ground True

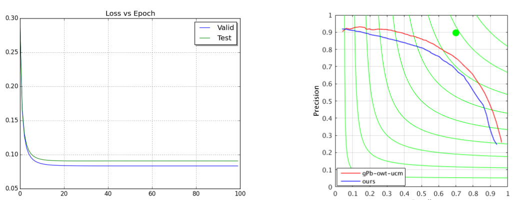

Figure 4. Loss vs Epoch curve(left) and Precision vs Recall curve(right)

read. This encouraged us to implement another file operation tools that allow user to convert the data into CPickle format by only two press.

4. Experiment

We evaluate our approach on the Berkeley Segmentation Dataset and Benchmark (BSDS 500)[1] which is composed of 200 training, 100 validation and 200 testing images. Each image is mannually annotated by an average of five humans with ground truth contours. Edge detection accuracy is evaluated using the precision recall curve, computed by the benchmark code provided by BSDS 500.

Figure 4 shows the learning curve and comparison with benchmark method. Though our method cannot beat baseline method, we still believe it can do better when there is extra computational resources. In this experiment, we were only able to use 32 filters for each convolution layer due to the image size and number of layers. The intermediate output can rapidly eat out memory and crash if there are more filters. Because of this reason, we were only able to train the model with less preferred configuration.

The result is still encouraging. Even though our method does not beat benchmark, the output of the network is reasonable. Figure 3 shows the boundaries detected by our implementation.

Figure 5. Performance drop when the image has complex background.

For simple images that does not have complex objects, our model can accurately detect the boundaries. And, our model also provide levels of details that can support later post processing to select from.

However, we also notice that, in images has noisy background, our model capture too much of the noise, which can be ignored by benchmark algorithms. We assume this is because our training is forcing the network to use the combined network but not selecting model through learning.

4.1. Conclusion

In this project, we exploited several boundary detection methods and tried to use novel convolution-deconvolutional neural network to solve the problem.

There were several challenges in this task. There is no existing work that applies deconvolutional structure on boundary detection. We found there is no existing code for achieving deconvolutional network in Theano[2][3] and even some popular extension software package of it like Keras[5]. This situation forced us to take lots of time on figuring out a practical method to do deconvolution that can run fast enough on GPU. Furthermore, the BSDS500 dataset was in the format of matlab, which is hard to access through python.

Those challenges motivated us to create one tiny but useful toolbox to help user establish deconvolutional network and apply boundary detection task on BSDS500 dataset.

The experiment result seems to be quite reasonable and within our expectations. It does not show competitive result to the state of art methods. But We believe this is because our implementation and configuration is still not optimal and requires tuning.

Due to time limitation, we were not able to implement an advanced convolution-deconvolutional network that supports automatic layer configuration. However, we still delivered the idea that we proposed at project beginning in this report. and We believe this novel idea can work better. This will still be our future work.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[4] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[5] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[6] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

[8] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[9] 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*, pages 580–587, 2014.

[10] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Computer vision–ECCV 2014*, pages 297–312. Springer, 2014.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] J. Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1(1):37–42, 1983.

[13] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(1):57–74, 2003.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[16] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1713–1721, 2015.

[17] X. Ren. Multi-scale improves boundary detection in natural images. In *Computer Vision–ECCV 2008*, pages 533–545. Springer, 2008.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[19] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2368–2376, 2015.

[20] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[21] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025, Nov 2011.